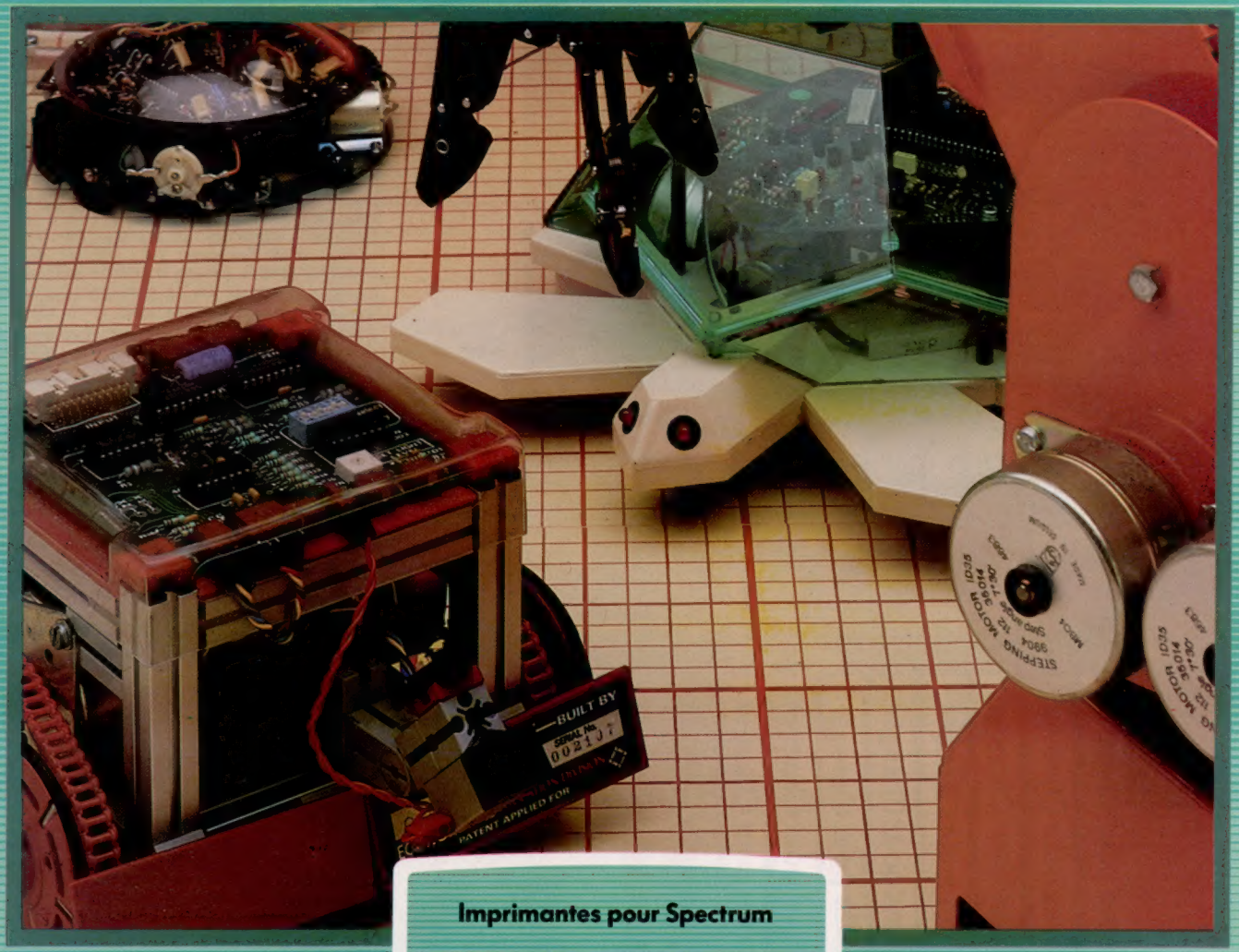
ABC INFORMATIQUE

N° 88

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE



- Imprimantes pour Spectrum
- Le CP/M au contrôle
- Le prolog
- Les voix de la SNCF

EDITIONS ATLAS

M 6062-88-12,50 F 90 FB - 3,80 FS - $ 1.95

Page manquante
(publicité et colophon)

Quelle impression?

Les interfaces du Spectrum ne permettent pas d'utiliser la plupart des imprimantes. Voici néanmoins quatre solutions avantageuses par rapport à la ZX.

Quatre solutions
Au cours des dernières années, de nombreuses imprimantes thermiques sont apparues sur le marché. Bien que certaines machines, comme la Floyd 40 et l'Alphacom 32, soient dédiées au Spectrum, d'autres, comme l'Epson P40 ou la Brother HR-5, sont proposées pour de nombreux micros domestiques. (Cl. Chris Stevens.)

L'imprimante est l'une des premières unités périphériques dont les possesseurs d'ordinateurs envisagent l'achat. Mais ceux qui possèdent un Spectrum ont à faire face à de nombreuses difficultés en ce domaine. D'une part, cet ordinateur ne possède aucune des interfaces conventionnelles utilisées par les imprimantes. D'autre part, l'imprimante proposée par Sinclair Research, dont la production est interrompue, a la réputation de produire des caractères d'une qualité médiocre et dont la lisibilité diminue progressivement. Puisque Sinclair a décidé de ne pas proposer de version améliorée, d'autres constructeurs ont produit de leur côté des imprimantes thermiques à bas prix.

Il faut tenir compte de plusieurs éléments essentiels lors de l'achat d'une imprimante. Si le coût est un facteur important, il peut exister des frais cachés, qui n'apparaissent pas dans le prix d'achat de la machine. Un fabricant peut déclarer qu'une imprimante est conçue pour un Spectrum alors qu'elle nécessite en fait l'achat d'une interface RS232C. Par conséquent, avant de pouvoir utiliser l'imprimante, une extension d'Interface 1 (qui connecte une prise RS232C au Spectrum) doit également être achetée.

Frais cachés

Le papier peut également entraîner des frais supplémentaires. De nombreuses imprimantes n'utilisent que du papier conçu spécialement pour elles. Par exemple, les imprimantes thermiques nécessitent un papier thermosensible spécial. Ainsi, après avoir payé votre imprimante, vous serez effectivement lié au bon vouloir du fabricant, et le papier peut devenir difficile à trouver et très coûteux.

Lors de l'achat d'une imprimante, il est donc essentiel de s'assurer que la boutique, qui vend l'unité, dispose d'un stock abondant et régulièrement approvisionné en papier et autres fournitures, et de connaître le coût exact de ces produits. Vous devriez également poser des questions au sujet de l'entretien. Les appareils comportant des pièces mobiles mécaniques sont plus sujets aux défaillances que les composants électro-

niques. Il est donc préférable d'acheter votre imprimante à une société qui possède un service après-vente efficace.

Le problème de la compatibilité logicielle ne doit pas non plus être négligé. Toutes les imprimantes sont programmées afin de répondre à des codes (généralement des caractères ASCII) qui lui commandent l'exécution de certaines fonctions comme un retour chariot, le nombre de caractères par ligne, la justification. Même s'il existe une certaine standardisation, à certaines fonctions peuvent correspondre des codes différents selon les machines. Il est donc toujours préférable de vérifier le bon fonctionnement de l'imprimante avec le Spectrum avant de l'acheter.

Les mêmes précautions s'appliquent à l'achat de tout logiciel. Il n'y a rien de plus frustrant que des codes de commande qui fonctionnent très bien à partir du BASIC, mais qui ne peuvent être utilisés à l'intérieur d'un programme de traitement de texte. Vous pouvez ainsi vous retrouver dans la situation peu enviable où vous devez sauvegarder votre copie sur bande ou sur disque, quitter le programme et lire la copie comme un fichier séquentiel afin de pouvoir l'imprimer.

Après avoir défini les critères de sélection qui doivent être retenus lors de l'achat d'une imprimante pour le Spectrum, examinons certaines des imprimantes thermiques peu coûteuses actuellement offertes sur le marché. Parmi les imprimantes présentées, la moins chère est l'Alphacom 32. Cette unité ressemble fort à l'imprimante que proposait Sinclair. Les polices de caractères utilisées ressemblent à celles de l'imprimante ZX et, comme cette dernière, elle ne peut imprimer que sur un maximum de 32 colonnes.

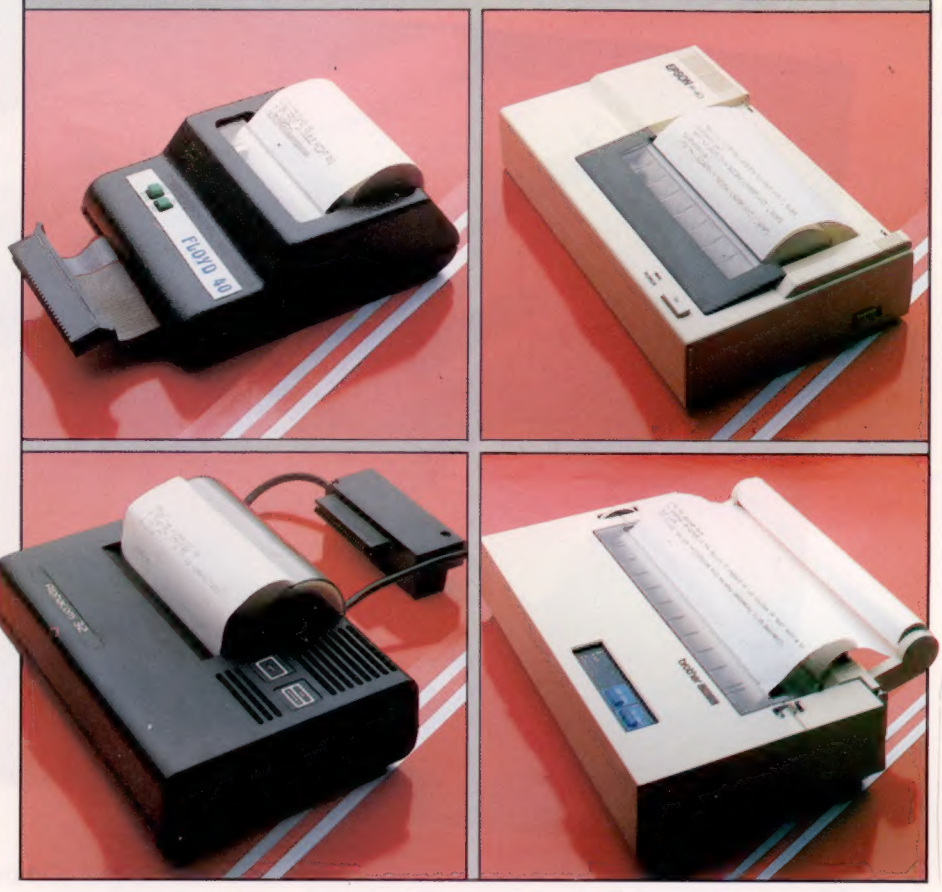
Pour une raison quelconque, les fabricants ont choisi de munir l'Alphacom de sa propre alimentation. La présence de ce boîtier additionnel et de ses câbles associés est inutile, puisque le Spectrum a une alimentation de 9 V, suffisante pour alimenter une imprimante thermique.

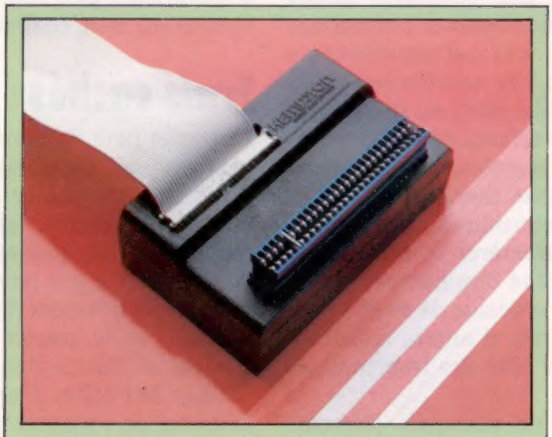
La construction de la Floyd 40, de Shiva Marketing, semble un peu douteuse. Le boîtier se plie facilement et le rouleau sur lequel repose le papier n'est qu'un mince axe de bois. Cela dit, la machine est déjà beaucoup mieux conçue que l'Alphacom.

L'alimentation provient du bus d'extension du Spectrum, et aucun boîtier additionnel n'est nécessaire. La Floyd 40 utilise aussi du papier blanc grâce auquel l'impression noire est beaucoup plus lisible. Ce qui distingue réellement cette machine de l'Alphacom et de l'imprimante ZX est le fait qu'elle accepte de nombreux caractères de commande.

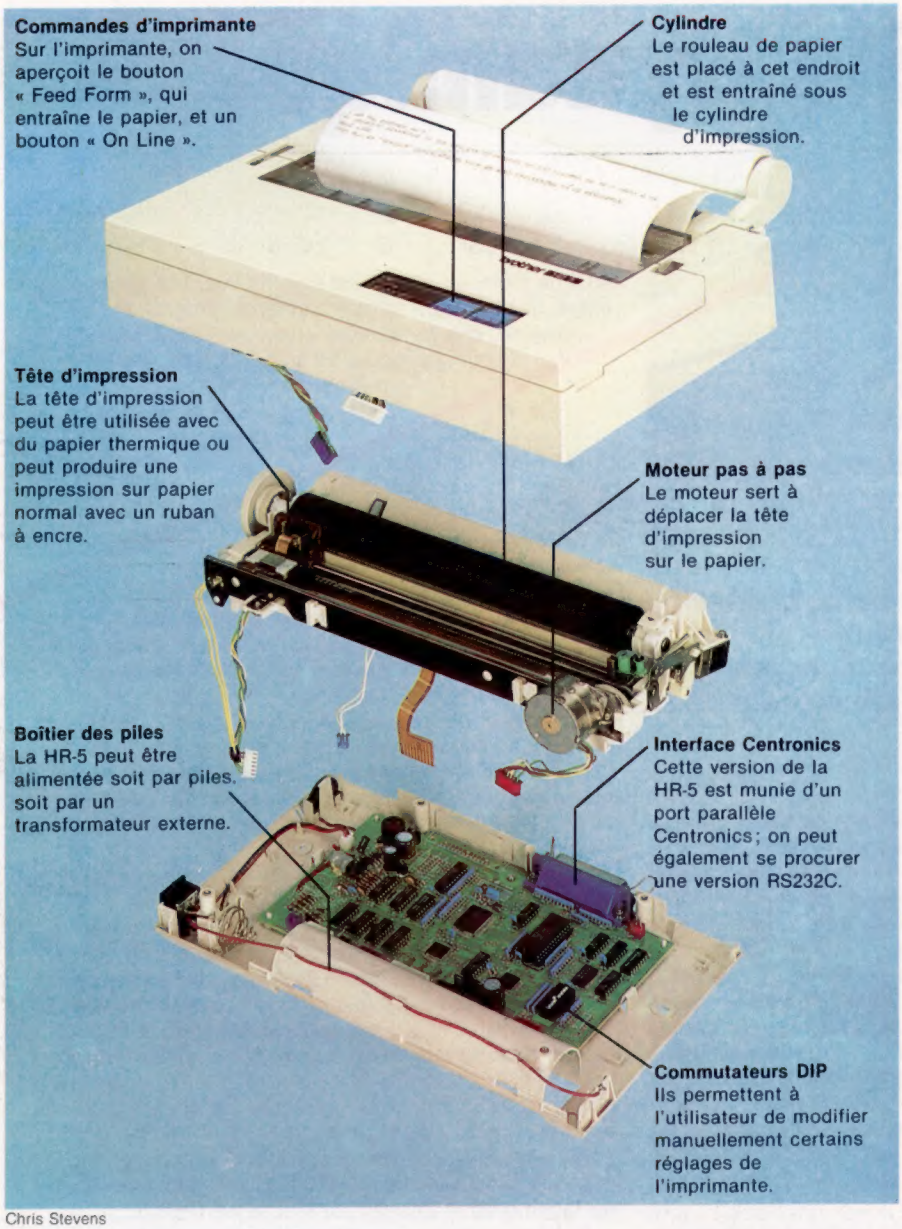
Ces caractères sont envoyés vers l'imprimante au moyen d'une commande LPRINT suivie de guillemets. Le caractère de commande lui-même est placé entre des points d'exclamation qui indiquent à l'imprimante de ne pas l'imprimer puisqu'il s'agit d'une commande. Par exemple, la ligne LPRINT«!H!» commande à l'imprimante Floyd 40 d'imprimer des caractères de double hauteur. L'émission de la commande une seconde fois annule la mise en page.

MODÈLE	Alphacom 32	Floyd 40	Epson P40	Brother HR5
PRIX	×	×	× ×	× × ×
INTERFACE UTILISÉE	Bus d'extension	Bus d'extension	RS232 ou parallèle Centronics	RS232 ou parallèle Centronics





Interface Kempston
Afin d'utiliser certaines des imprimantes parmi les plus perfectionnées (qui sont munies d'interfaces RS232C ou Centronics), le possesseur d'un Spectrum doit acheter une interface pour établir les bonnes connexions. La plus populaire de ces interfaces est la Kempston qui donne accès à une grande variété d'imprimantes compatibles Centronics.



Chris Stevens

D'autres fonctions permettent également de produire des caractères graphiques, un bouclage de mot, une impression double largeur ou inversée. La tête d'impression, de cinq points par sept, produit de très belles illustrations en mode graphique, et le format d'impression est comparable à celui produit par des machines coûtant le double. Outre le fait qu'elle est assez lente, le seul inconvénient réel de l'unité est la largeur limitée de son rouleau de papier (80 mm).

La troisième machine présentée est l'Epson P40 qui est vendue en deux versions de base correspondant aux principaux types d'interfaces d'imprimante. La P40S est une version série avec une prise RS232C, tandis que la P40P possède un port Centronics. Cependant, comme le Spectrum ne possède aucune de ces interfaces en version standard, vous devez acheter une Interface 1 ou l'une des autres interfaces existantes.

Autre problème, la prise RS232 fournie par Sinclair sur l'Interface 1 est un connecteur de type D à sept broches non standard. Les manuels de la P40 et de l'Interface 1 expliquent de quelle façon doivent être positionnées les broches, et il ne vous reste plus qu'à effectuer un petit travail de soudure. Si, cependant, vous ne pouvez vous imaginer utilisant un fer à souder, il est probablement préférable d'acheter l'une des interfaces proposées par des tiers. Kempston, par exemple, propose des interfaces qui se branchent dans le panneau arrière du Spectrum ainsi que les connexions standards destinées aux unités RS232 et Centronics.

L'Epson P40, malgré ses dimensions réduites, peut imprimer en mode 40 ou 80 colonnes. La machine peut également utiliser de nombreux codes d'échappement, qui sont disponibles sur les unités plus évoluées de la gamme. Bien sûr, le Spectrum n'a pas de touche « Escape », mais les codes peuvent être envoyés sous la forme de codes ASCII dans le format CHR$(27);«E»; (où CHR$(27) représente le code ASCII de Escape) qui indique à la P40 de passer en caractères gras.

La P40 peut également effectuer de nombreuses autres opérations, comme modifier le jeu de caractères, passer en mode image par bit (qui vous permet de créer vos propres caractères) et définir un mode condensé. En plus de ces modifications qui peuvent être obtenues par logiciel, il y a aussi des commutateurs DIP permettant de régler la parité et le nombre de colonnes imprimées.

Parce que la P40 fait partie de la grande famille Espon, vous pouvez être assuré d'un excellent service pour de nombreuses années. Le seul inconvénient de la machine est qu'elle coûte relativement cher; ainsi, après avoir acheté l'interface, le coût global de l'imprimante peut dépasser celui de l'ordinateur.

Contrairement aux autres imprimantes mentionnées dans cet article, la Brother HR-5 peut utiliser soit du papier thermique, soit du papier ordinaire si un ruban est installé. La machine offre un autre avantage par rapport aux autres imprimantes : elle peut accepter du papier A4 et peut être utilisée pour imprimer des lettres et autres applications standards de traitement de texte. La largeur additionnelle permet à la Brother d'imprimer jusqu'à 132 caractères par ligne.

Comme l'Epson, la HR-5 utilise les codes Escape pour sa mise en page, et est donc capable de produire une grande variété de caractères. Son fonctionnement est presque silencieux. Malheureusement, comme l'Epson, la machine est munie d'une interface parallèle Centronics ou d'une interface RS232C. Vous devez donc de nouveau acheter une interface appropriée pour utiliser l'imprimante à partir d'un Spectrum. Mais la qualité d'impression est excellente.

Lors du choix d'une imprimante, les problèmes deviennent plus sérieux si vous décidez d'acheter une machine dont les capacités dépassent celles des imprimantes conçues spécialement pour le Spectrum. L'Alphacom et la Floyd 40 offrent l'avantage de pouvoir être simplement branchées à l'ordinateur et d'accepter des commandes normales du Spectrum comme COPY mais en se limitant à l'impression de listages.

Contrôle dynamique

Cet article est le premier d'une série qui étudiera en détail la création, la structure et la mise en œuvre d'un des systèmes d'exploitation les plus connus — CP/M.

Lorsque les mémoires de masse à disquettes furent inventées au début des années soixante-dix, il fut très vite évident que les usagers auraient besoin d'un système d'exploitation capable de gérer les informations qu'elles contenaient, sans être obligés de tout faire « à la main ». A peu près à la même époque, la firme américaine Intel, productrice de microprocesseurs, décida de mettre au point un système de ce genre, destiné à sa nouvelle série de puces (la famille des 8000). Son équipe de recherche était alors dirigée par un jeune ingénieur-consultant nommé Gary Kildall. Des travaux entrepris naquit un programme qui se fit vite connaître sous le nom de CP/M (au départ acronyme de Control Program/ Monitor, auquel on substitua plus tard Control Program for Microprocessors).

Ce logiciel connut un succès foudroyant dès sa sortie en 1975. Ce fut même un tel triomphe que lorsqu'un groupe de concepteurs quitta Intel pour fonder une nouvelle compagnie nommée Zilog Inc., tous décidèrent de rendre leur premier produit — le microprocesseur Z80 — compatible avec le 8008, de façon que lui aussi puisse accepter CP/M. De son côté, Kildall créa également sa propre firme, Digital Research, qui est aujourd'hui l'une des plus grosses sociétés productrices de logiciels du monde.

CP/M est actuellement un standard de fait pour les micro-ordinateurs 8 bits, bien, qu'à l'époque, on ait vu apparaître de nombreux systèmes rivaux, en particulier ceux conçus autour du microprocesseur 6502 (qui est incompatible avec CP/M). Cette réussite inégalée est d'autant plus impressionnante que le programme a près de dix ans d'âge — ce qui, en informatique, correspond à peu près à la préhistoire.

Pour pouvoir en faire usage, vous aurez besoin d'un ordinateur comportant une puce 8008 (ou un modèle compatible), d'un lecteur de disquettes et d'au moins 48 K de mémoire vive. C'est en effet l'espace minimal exigé, dès lors qu'il faille accueillir le programme lui-même, ses fichiers de commande (qui peuvent occuper jusqu'à 8 K chacun) et les fichiers sur disquettes qu'il aura à manipuler. La récente apparition des micro-ordinateurs 16 bits marquera sans doute la fin de CP/M, du moins dans le domaine de la gestion informatisée. Pourtant, il a été adopté par de nombreux constructeurs d'ordinateurs domestiques équipés d'un Z80 : leurs appareils peuvent de cette façon échapper aux jeux vidéo et proposer un certain nombre d'applications « sérieuses ». De surcroît, l'usager disposant du logiciel et d'un lecteur de disquettes peut instantanément accéder à un très grand nombre de logiciels rédigés au cours des années soixante-dix pour les ordinateurs de gestion construits autour du 8008 ou du Z80. Le gros problème auquel toute nouvelle machine doit faire face — l'absence de programmes — est ainsi contourné. Cette stratégie fut celle de bon nombre de fabricants de micro-ordinateurs.

La plupart des ordinateurs personnels disposent d'un système d'exploitation intégré (activé dès qu'ils sont mis sous tension). Mais CP/M est presque toujours installé sur disquette (c'est ce qu'on appelle un « disque système »), et chargé après initialisation. Il n'y a pas de raison particulière à cela, et certaines firmes proposent d'ailleurs le programme sur ROM. Sans doute est-ce un héritage historique : la mémoire des premiers ordinateurs était très limitée ; si CP/M y avait été implanté en permanence, il aurait pris beaucoup trop de place. Aussi préféra-t-on le laisser sur disquette, de façon à ne le charger qu'en cas de besoin.

En règle générale, un disque système CP/M comprend le programme lui-même, ainsi que d'autres, de taille plus réduite, qu'on peut appeler par son intermédiaire, et qui permettent l'exécution de commandes spécifiques concernant la gestion des disquettes. Toutefois, CP/M est un système d'exploitation complet, et non un simple gestionnaire de fichiers.

Une fois chargé en mémoire, le logiciel prend le contrôle de l'ordinateur, dont la ROM est dépossédée de son rôle habituel de gestionnaire de l'ensemble. C'est ainsi qu'une commande BASIC quelconque, qui serait normalement exécutée par la machine, ne sera pas reconnue par le CP/M et provoquera l'apparition d'un message d'erreur. A dire vrai, aucun programme BASIC ne peut tourner sous CP/M, sauf si celui-ci s'est vu adjoindre un interpréteur ou un compilateur.

Lors de la mise sous tension, le disque système CP/M est généralement lancé automatiquement, et un répertoire des commandes disponibles est affiché à l'écran (voir illustration). L'examiner de près permet d'apprendre beaucoup de choses sur la structure du

programme lui-même. Les commandes (comme tous les fichiers de CP/M) sont en deux parties. Il y a d'abord le nom « primaire ». C'est lui que nous taperons si nous avons besoin d'un fichier commande particulier, avec le nom du fichier sur lequel nous voulons travailler : le tout sera chargé et lancé automatiquement.

On appelle « extension » la seconde partie du nom (après le point). Elle permet à CP/M (et à vous, par la même occasion) de savoir à quel type de fichier il a affaire. Ceux qui se trouvent sur le disque système sont tous des fichiers de commande; ils sont donc suivis de l'extension COM. Mais il en existe d'autres que nous étudierons ultérieurement.

Autre détail à noter quand on examine le répertoire de la disquette : CP/M lui-même n'y apparaît pas. On pourrait donc penser, à première vue, qu'il se réduit en fait à un ensemble de fichiers de commande de petites dimensions. Ce n'est pas tout à fait faux, mais, si l'on y regarde de plus près, on constate que si certaines commandes fondamentales (LOAD, par exemple) sont bel et bien là, d'autres (ainsi SAVE) ne sont pas mentionnées. Il est donc évident que le répertoire ne nous dit pas tout.

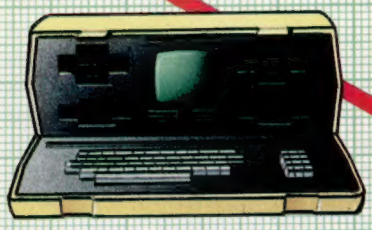
```
WRENMENU version 1.1
(c) 1984 Quantec Systems and Software Ltd
A>dir
A: WRENMENU COM : CPM3    SYS : COPYSYS  COM : ERASE   COM : SET     COM
A: SHOW     COM : SUBMIT  COM : TYPE     COM : DIR     COM : GET     COM
A: PUT      COM : RENAME  COM : SETDEF   COM : PIP     COM : HELP    COM
A: HELP     HLP : FC      COM : DEVICE   COM : FORMAT  SUB : FC      GET
A: FORMAT   GET : CONFIGUR COM
A>
```

Coup d'œil au répertoire
Taper DIR provoque l'affichage d'une liste des fichiers contenus sur le disque système CP/M. A moins que vous n'y ayez déjà ajouté d'autres choses, tous ceux que vous verrez mentionnés seront suivis du suffixe .COM, qui indique que l'on a affaire à des fichiers de COMmande. Le répertoire donne par ailleurs des précisions supplémentaires : espace mémoire occupé par chacun d'eux, nom du fichier à partir duquel vous avez accédé au répertoire, etc.

Il fut donc décidé d'en laisser une partie sur disquette, de sorte qu'elles ne seraient appelées qu'en cas de besoin, permettant ainsi d'économiser de l'espace mémoire.

Par ailleurs, une telle méthode a certains avantages. Des commandes transitoires supplémentaires peuvent être ajoutées très facilement à l'ensemble, et il suffira pour cela de les insérer dans le répertoire, et de les faire suivre d'une extension COM. Cette remarquable souplesse d'emploi — CP/M est comme un cadre général, à l'intérieur duquel on peut placer ce qu'on veut — a eu pour résultat la création d'un nombre énorme de programmes d'application. C'est un très bel exemple d'effet « boule de neige ». C'est ainsi que le célèbre logiciel de traitement de texte Wordstar (bien qu'il soit traité comme un programme séparé qui peut tourner sous CP/M) n'est considéré par ce dernier que comme un de ses fichiers de commande lors de son chargement.

Osborne 1



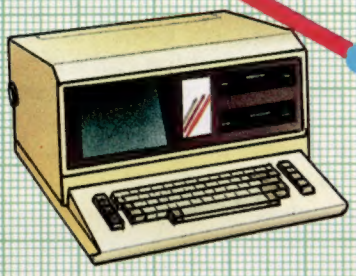
Nous avons déjà fait remarquer qu'un système d'exploitation se doit d'être aussi « transparent » que possible, de façon à exiger le moins possible de vous, donc d'assurer un bon fonctionnement de l'ensemble. Quand nous chargeons le disque système, CP/M est placé en RAM et y demeure en permanence tant que l'ordinateur n'est pas mis hors tension.

En même temps, un certain nombre de fichiers de commande d'emploi fréquent (on les appelle programmes utilitaires) sont chargés au même endroit avec le programme principal. Il devient donc inutile, pour en faire usage, de les appeler séparément à partir de la disquette.

Tout cela peut paraître un peu ambigu. Comment distinguer les deux types de commandes? Il faut recourir à une distinction entre commandes « intégrées » (invisibles sur le répertoire) et commandes « transitoires » (disponibles sur la disquette). Ces dernières, une fois exécutées, sont éliminées par le système d'exploitation et doivent être rechargées si l'on veut les utiliser de nouveau.

Le système peut paraître un peu arbitraire. Mais il a une double justification. D'abord les limitations du matériel lui-même : le CP/M est conçu pour tourner sur des micro-ordinateurs 8 bits, dont l'unité centrale ne peut adresser directement qu'un maximum de 64 K. Si toutes les commandes étaient chargées simultanément, l'usager ne disposerait plus que d'un espace RAM des plus réduits.

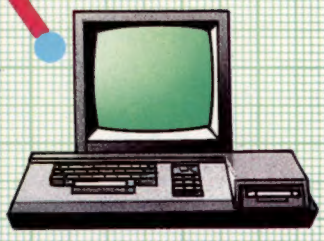
Wren



Cela a plusieurs avantages. Wordstar peut, par exemple, être mis en œuvre sur tout ordinateur fonctionnant sous CP/M; il n'a pas besoin de commandes à l'unité de disquettes spécifiques, puisqu'il utilise celles du programme principal. Cela signifie en particulier qu'il est de taille beaucoup plus réduite, ce qui libère de l'espace mémoire dont on peut se servir pour des fichiers texte.

Nous venons de voir comment CP/M a été créé, et quelle est sa structure de base. Le prochain article traitera de certains des programmes utilitaires disponibles, et étudiera leur fonctionnement.

Amstrad CPC 664



1981 1982 1983 1984 1985

Échanges fructueux

Si vous n'avez commis aucune erreur grave, vous voilà dans le Nouveau Monde. Ayez un peu le sens des affaires pour revenir confortablement en Europe.

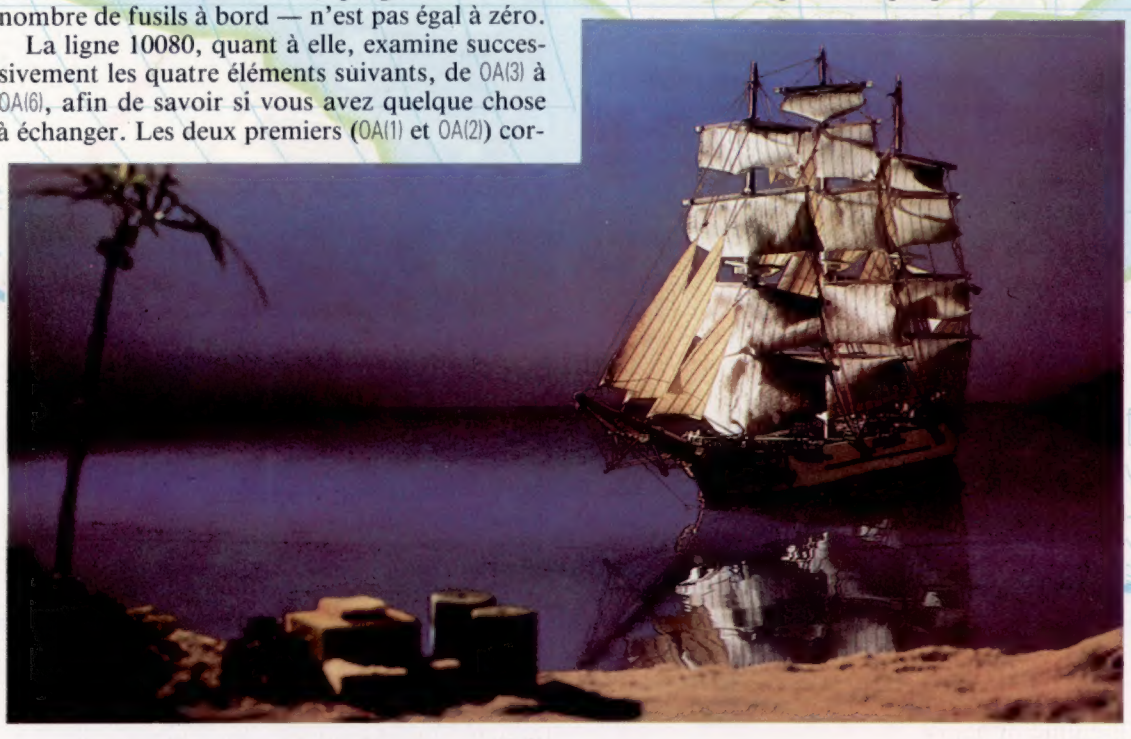
La ligne 892 du programme principal appelle le sous-programme consacré aux échanges avec les indigènes. Sa première tâche consistera à voir si vous avez ou non amené des fusils avec vous. Ils peuvent, bien sûr, se révéler utiles — pour repousser des pirates, par exemple — mais le chef des habitants du Nouveau Monde a pour eux une profonde aversion, qui s'explique aisément... Cela signifie malheureusement que, si vous êtes venu armé lors des opérations de troc, il sera très mécontent et refusera de les échanger contre quoi que ce soit. Les autres marchandises, quant à elles, seront réparties en lots à négocier.

Le sous-programme vous informe ensuite du prix — fixé avant votre départ — de chacune des choses proposées par le chef. Le sel est échangé en premier, contre des perles, des statues ou des épices. Vous procéderez de la même manière avec les ballots de tissu, les couteaux et les bijoux. Bien entendu, le chef n'aura pas à préciser au capitaine qu'il ne désire pas de fusils s'il n'y en a pas à bord ! C'est bien pourquoi la ligne 10072 vérifie le tableau des provisions, OA, et s'assure que son second élément — OA(2), qui garde trace du nombre de fusils à bord — n'est pas égal à zéro.

La ligne 10080, quant à elle, examine successivement les quatre éléments suivants, de OA(3) à OA(6), afin de savoir si vous avez quelque chose à échanger. Les deux premiers (OA(1) et OA(2)) correspondent aux médicaments et aux fusils ; ils ne peuvent donner lieu à un troc et ne sont donc pas pris en compte. S'il vous reste des couteaux, du sel, du tissu ou des bijoux, vous serez averti que le chef est désireux de les acquérir contre des perles, des statues ou des épices. Si vous n'avez rien à offrir (les quatre éléments du tableau auront alors une valeur de zéro), la partie prendra fin, faute d'échanges possibles.

L'ordinateur vous informe ensuite de la valeur de chacune des marchandises proposées en vous avertissant qu'elle a pu changer entre-temps. Une boucle est ensuite créée lignes 10130 à 10200, afin de gérer les échanges. Il s'agit avant tout de calculer la quantité de perles, de statues et d'épices offertes par le chef pour chaque lot de marchandises que vous souhaitez lui présenter. Le programme vous demande alors ce que vous voulez acquérir en échange, puis il charge les produits sur votre navire.

La boucle va de 3 à 6, ce qui lui permet d'ignorer les fusils et les médicaments (qui sont les deux premiers éléments du tableau OA(I)), puis s'occupe séparément de chaque marchandise. La ligne 10135 voit si la valeur de chaque élément du tableau est ou non égale à zéro et, si c'est le cas, passe à l'élément suivant du tableau OA en sautant directement à l'instruction NEXT de la ligne 10200. Dans le cas contraire, vous saurez de quelle quantité vous disposez : elle sera affichée par la ligne 10145. Les lignes 10150 à 10153 assurent l'affichage des différents types de marchandises, en fonction de la valeur de T. Le chef ayant fait une première offre de perles, de statues et d'épices, le programme effectue



des calculs pour savoir exactement combien vous pourrez en recevoir, vu ce que vous proposez, et cela pour chacune de vos marchandises. Il fait alors usage du tableau à deux dimensions EQ(,) créé ligne 63, qui gère les taux d'échange.

L'expression contenue ligne 10165 permet de déterminer la quantité de perles offertes pour chacun des articles de la boucle. Elle multiplie la quantité des marchandises que vous avez à échanger par le taux d'échange tel qu'il est défini par EQ(,). La boucle allant de 3 à 6, il faut soustraire 2 à T pour le faire correspondre au numéro des éléments des provisions de EQ(,), qui vont de 1 à 4. Si, par exemple, la boucle prend en compte le sel, T, le compteur de boucle sera égal à 3; EQ(T-2,1) permet donc de mettre en œuvre une intersection entre le premier élément du premier indice (donc le sel) avec le premier élément du second indice (les perles). EQ(,) est alors égal à 5. Pour le sel, la ligne 10165 multipliera donc la quantité de sel par 5 pour savoir combien de perles seront proposées en échange.

La ligne 10166 procède de la même façon à propos des statues offertes par le chef. Le taux d'échange correspondant est conservé par le second élément du second indice dans le tableau EQ(,). Chaque valeur successive du compteur de boucle T envoie donc le programme vers la section appropriée de la seconde colonne du tableau. La ligne 10167 fait de même avec les épices, mais en se servant de la troisième colonne du tableau des taux d'échange.

Après avoir appris ce que le chef entend échanger, vous devrez décider de ce que vous allez accepter dans ses propositions. Vous aurez donc à taper un nombre compris entre 1 et 3, correspondant à votre choix. La ligne 10176 vérifie la validité de votre réponse en s'assurant qu'elle est bien conforme (elle ne doit pas être inférieure à 1 ou supérieure à 3). Les marchandises acquises sont ensuite amenées à bord, et la ligne 10180 en inscrit la quantité dans le tableau AO(), DIMensionné ligne 68.

AO() a trois éléments, qui ne sont autres que les trois types de marchandises offertes par le chef. Pour calculer les quantités exactes de ce que vous emporterez, le tableau des taux d'échange, EQ(,), est mis en œuvre. L'élément correspondant est multiplié par la quantité de marchandises échangées, OA(T). I est le nombre que vous devez taper en fonction de votre choix d'une certaine catégorie d'objets; il permet de sélectionner le bon élément de AO() : perles, statues ou épices. I sert aussi de second indice de EQ(,).

Notez que si, plus tard, de nouvelles quantités de perles, de statues ou d'épices sont acquises lors de marchandages ultérieurs, elles devront être ajoutées à celles déjà détenues par le tableau AO(). Cela étant fait, la boucle reprend au tout début si c'est nécessaire.

Les palabres cessent quand les deux parties ont échangé tout ce qui était possible de l'être. Les lignes 10220 et 10244 affichent la quantité de chaque marchandise acquise, contenue dans le tableau AO(). Le contrôle repasse alors au programme principal.

Module douze : les échanges

Addition au programme principal

```
892 GOSUB10070
```

S/P Échanges

```
10070 PRINTCHR$(147):GOSUB9200:REM TRADING
10072 IFOA(2)=0THEN10080
10074 S$="LE CHEF NE VEUT PAS=":GOSUB9100
10075 S$="DE VOS FUSILS - TROP DANGEREUX=":GOSUB91
00
10078 PRINT:GOSUB9200
10080 IFOA(3)<>0ORDA(4)<>0ORDA(5)<>0ORDA(6)<>0THEN
10100
10085 S$="IL NE VOUS RESTE RIEN=":GOSUB9100
10090 S$="A ECHANGER=":GOSUB9100
10095 GOTO10038
10100 S$="EN ECHANGE DES COUTEAUX=":GOSUB9100
10102 S$="DU SEL, DU TISSU OU DES BIJOUX=":GOSUB910
0
10104 S$="IL VOUS PROPOSE DES PERLES, DES STATUES=":GOSUB910
0
10106 S$="ET DES EPICES=":GOSUB9100
10108 PRINT:GOSUB9200
10110 S$="A VOTRE DEPART=":GOSUB910
0
10112 S$="ELLES VALAIENT:=":GOSUB9100
10114 S$="PERLES : 2 PCS D'OR CHAQUE=":GOSUB9100
10116 S$="STATUES : 2 PCS D'OR CHAQUE=":GOSUB9100
10118 S$="EPICES : 1 PC D'OR LE GRAMME=":GOSUB9100
10120 PRINT:GOSUB9200
10122 S$="MAIS LEUR VALEUR PEUT CHANGER=":GOSUB9100
10124 S$="D'ICI VOTRE RETOUR=":GOSUB9100
10125 PRINT:GOSUB9200:S$=K$:GOSUB9100
10126 GET I$:IFI$=""THEN10126
10130 FORT=3TO6
10135 IFOA(T)=0THEN10200
10140 PRINTCHR$(147):GOSUB9200
10145 PRINT "VOUS AVEZ":OA(T);
10150 IFT=3THENS$="SACS DE SEL="
10151 IFT=4THENS$="BALLOTS DE TISSU="
10152 IFT=5THENS$="COUTEAUX="
10153 IFT=6THENS$="BIJOUX="
10155 GOSUB9100
10156 PRINT:GOSUB9200
10160 S$="EN ECHANGE LE CHEF PROPOSE :=":GOSUB91
00
10165 PRINT"EITHER":OA(T)*EQ(T-2,1);"PERLES"
10166 PRINT"    OR":OA(T)*EQ(T-2,2);"STATUES"
10167 PRINT"    OR":OA(T)*EQ(T-2,3);"GRAMMES D'EPICES"
10168 PRINT:GOSUB9200
10170 S$="VOULEZ-VOUS DES PERLES, DES STATUES=":
GOSUB9100
10172 S$="OU DES EPICES ?=":GOSUB9100
10174 S$="(ENTREZ 1,2 ou 3)=":GOSUB9100
10175 INPUTI$
10176 I=VAL(I$):IFI<1ORI>3THEN10174
10180 AO(I)=AO(I)+(OA(T)*EQ(T-2,I))
10190 PRINT:PRINT"LES" ;T$(I); "SONT MONTEES A BORD"
10192 S$=K$:GOSUB9100
10194 GETI$:IFI$=""THEN10194
10200 NEXT
10210 PRINT:PRINT:GOSUB9200
10215 S$="      FIN DES ECHANGES=":GOSUB9100
10216 PRINT:GOSUB9200
10218 S$="VOUS AVEZ DESORMAIS:=":GOSUB9100
10220 PRINTAO(1);"PERLES"
10222 PRINTAO(2);"STATUES"
10224 PRINTAO(3);"GRAMMES D'EPICES"
10226 PRINT:GOSUB9200
10228 S$=K$:GOSUB9100
10229 GETI$:IFI$=""THEN10229
10230 RETURN
```

Variantes de basic :

Spectrum :

Remplacez tout au long EQ(,) par Q(,), AO() par E(), et procédez aux modifications suivantes :

```
10070 CLS
10126 LET I$=INKEY$:IF I$="" THEN GOTO 10126
10140 CLS:GOSUB 9200
10229 LET I$=INKEY$:IF I$="" THEN GOTO 10299
```

BBC Micro :

Procédez aux modifications suivantes :

```
10070 CLS
10126 I$=GET$
10140 CLS:GOSUB 9200
10229 I$=GET$
```

Stratégie

Nous poursuivons notre étude sur l'intelligence artificielle par l'élaboration de programmes susceptibles de prévoir le déroulement d'une partie d'échecs et de décider des mouvements.

Création McConachie en association avec Chris Tucker

Grand Maître
La plupart des programmes de jeu d'échecs sont capables d'anticiper un certain nombre de coups pour choisir le meilleur mouvement. L'ordinateur utilise pour cela un arbre logique d'exploration des coups possibles. Dans les premiers temps de l'intelligence artificielle, qu'un ordinateur sache jouer aux échecs était considéré comme la démonstration de l'intelligence d'une machine.

Les jeux sur ordinateurs évoquent généralement des scènes de vaisseaux intersidéraux ou de créatures caverneuses. Mais il n'en a pas toujours été ainsi. Dans les premiers temps de l'informatique, les pionniers cherchaient à programmer un ordinateur pour jouer aux échecs.

L'écriture d'un tel programme était considérée comme le meilleur moyen pour tester l'intelligence de l'ordinateur. Il existe aujourd'hui des systèmes informatiques qui atteignent le niveau des meilleurs joueurs d'échecs internationaux. Cela dit, on peut difficilement affirmer que ces machines « pensent ». Les programmes de jeux d'échecs et autres « jeux mentaux » constituent néanmoins un terrain d'essai idéal pour les théories de planification stratégique.

La plupart des programmes de jeux, dits d'intelligence, font intervenir des techniques de recherches arborescentes qui tiennent compte, bien entendu, de l'adversaire. L'idée fondamentale est celle de l'anticipation. Le programme construit le jeu en forme d'arbre, prenant en considération ses propres mouvements, les réponses possibles de l'adversaire, les répliques, etc.

Le diagramme de jeu sous forme d'arbre représente l'arborescence des mouvements possibles pour un jeu imaginaire à deux personnes. La racine de l'arbre est la position en cours, avec MAX prêt à jouer. Les nœuds terminaux, ou feuilles, sont les positions de fin de jeu. L'arbre sert à choisir le meilleur mouvement, selon une procédure appelée « minimax ». Cette dernière a été exposée pour la première fois en 1949 par Claude Shannon. Sa méthode est d'assigner d'abord des valeurs numériques aux nœuds terminaux, par exemple 1 pour gagner, 0 pour une partie nulle et −1 pour perdre. Ces valeurs se combinent ensuite lors de la construction de l'arbre. On suppose que le joueur (MAX) prend toujours les plus grandes valeurs, le joueur (MIN) les plus petites.

Dans cet exemple, la valeur pour la racine est 0, signifiant ainsi que la partie est nulle (à condition que les deux joueurs ne fassent pas d'erreurs). Le meilleur mouvement en haut de l'arbre est donc M1, M3, ou M4, mais pas M2. Les règles de branchement et de calcul des valeurs de nœuds dépendent des règles spécifiques du jeu. Seuls les jeux très faciles, tels que le « morpion » ou les croix, permettent de tracer la totalité de l'arbre. Le jeu d'échecs, par exemple, a un « facteur de branchement » de 32. Cela signifie qu'il y a, approximativement, 32 mouvements possibles en tous points. Envisager les jeux au-delà de deux mouvements pour chaque joueur aboutirait à plus d'un million de nœuds terminaux. Cette explosion de combinaisons veut dire que les programmes d'échecs ne peuvent élaborer le jeu jusqu'à sa fin.

La plupart des programmes de jeu planifient aussi loin que possible et évaluent les positions trouvées. A cette fin, ils font une estimation de la valeur stratégique des positions « nœuds », même en l'absence du résultat réel. C'est ce qu'on appelle généralement la fonction d'évaluation statistique. Cette dernière introduit nécessairement un certain degré d'imprécision, du fait qu'il ne s'agit que d'une estimation du résultat final. Ce raisonnement d'anticipation et d'évaluation par une fonction imprécise est néanmoins une meilleure estimation que l'absence de recherche. En outre, il est censé être utilisé relativement près de la fin.

Pour prendre l'exemple du jeu de dames, nous pourrions mettre en place une fonction à quatre états :

D : avantage à la dame.
P : avantage à une pièce.
M : différence de mobilité.
C : contrôle au centre.

Ces attributs peuvent être évalués en examinant l'échiquier. Par exemple D = DD − DA (DD : dames en défense, DA : dames adverses).

On peut faire un certain nombre de constatations : il est préférable d'avoir plus de pièces que l'adversaire (le perdant n'en ayant plus); il est préférable de disposer de plusieurs mouvements possibles; enfin, le centre de l'échiquier, comme au jeu d'échecs, est la position la plus stratégique. Le programme doit, d'une façon ou d'une autre, combiner ces facteurs en un calcul général.

Si nous estimons qu'une dame vaut trois pièces ordinaires, qu'une prise demande deux mouvements et demi, qu'un mouvement gratuit vaut deux accès à une position centrale, notre fonction d'évaluation sera :

$$V = 15D + 5P + 2M + C$$

La pondération avec des entiers est généralement adoptée pour accélérer les calculs.

Cette fonction d'évaluation est tout à fait fruste. En comparaison, le programme classique d'Arthur Samuel, au début des années soixante, mettait en œuvre jusqu'à vingt-cinq paramètres. Les coefficients étaient également assez arbitraires. Une partie du plaisir du développement d'un tel programme tient à l'affinage de ces coefficients, la réalisation d'un équilibre entre eux. Un des points forts du programme de Samuel était qu'il ajustait automatiquement ses coefficients de pondération, ce qui constituait une forme rudimentaire d'acquisition de connaissances.

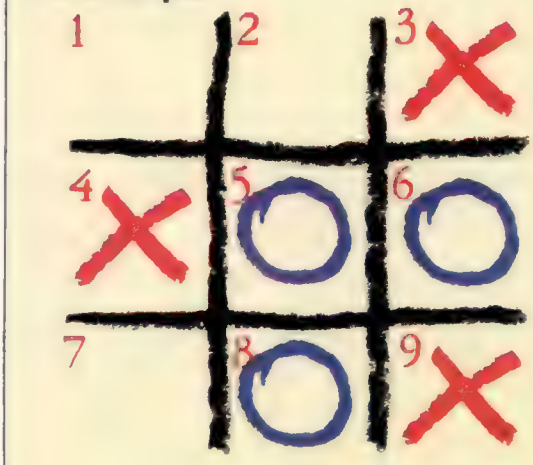
L'idée d'attribuer des valeurs numériques aux paramètres d'un jeu et de les combiner sous la forme d'une évaluation pondérée des positions s'est révélée très efficace au cours de ces trente dernières années. La fonction d'évaluation joue un rôle semblable à celui de la mesure heuristique d'une distance.

Un programme qui ne ferait qu'anticiper et évaluer les nœuds terminaux trouvés ne pourrait que rencontrer des difficultés. Cela est dû au fait que certaines positions de jeu sont stables, alors que d'autres sont instables. Au jeu d'échecs, l'état du jeu après une prise a toutes les chances d'être instable. Une contre-prise est susceptible d'avoir lieu au mouvement suivant. Lorsque cela se produit un coup après l'estimation du programme, l'évaluation est sérieusement faussée.

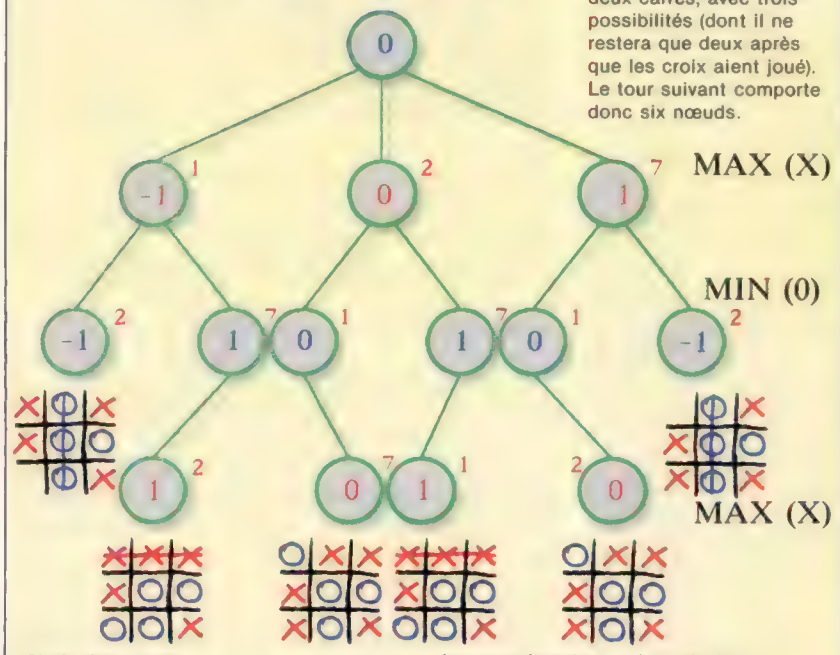
Pour pallier cette difficulté, la plupart des programmes n'ont pas un champ d'anticipation fixe. Ils disposent d'une caractéristique d'appréciation du degré de stabilité du jeu. Cela permet de savoir si une position peut être évaluée avec fiabilité. Dans le cas contraire, la recherche est poussée plus avant. Aux dames et aux échecs, cela suppose d'envisager des séquences longues.

L'algorithme « alpha-bêta » apparut pour la première fois en 1967 dans le programme Mack Hack de Greenblatt. Il s'agit d'un affinement de la méthode « minimaxi », qui donne le même résultat avec moins d'efforts. Le diagramme de la page suivante montre une partie de l'arbre du jeu entre deux joueurs appelés MINI et MAXI. La lettre apposée à chaque nœud (de A à L) donne l'ordre selon lequel l'arbre doit être passé en revue, utilisant une procédure verticale. Les nombres représentent les évaluations. Les embranchements marqués d'un trait sont des « tailles alpha », ceux marqués de deux traits, des « tailles bêta ». Il s'agit d'éliminer (de tailler l'arbre) les embranchements sans influence sur le résultat final. Une taille « alpha » se produit au nœud E, qui n'a donc pas besoin d'être évalué (ni ses éventuels descendants). Lorsque nous parvenons en E, nous savons que le nœud C obtient un score de 15, l'adversaire pouvant ramener ce dernier à 10 en D. Il est sans intérêt de chercher à savoir si l'adversaire peut nous faire perdre encore plus de points, le chemin passant par C étant bien plus intéressant. Ainsi, les autres descendants de F peuvent être éliminés.

L'état du jeu



Ce diagramme montre l'état en cours d'un jeu de morpions après six coups (les croix devant jouer). Il est possible d'établir une arborescence simple montrant les trois états terminaux du jeu. Le déploiement de cet arbre de jeu s'obtient en envisageant à chaque niveau le nombre d'options possibles. Au premier coup, trois nœuds logiques pour trois déplacements possibles (carrés 1, 2 ou 7), pour les croix. Lorsque c'est aux ronds de jouer, il ne reste que deux carrés, avec trois possibilités (dont il ne restera que deux après que les croix aient joué). Le tour suivant comporte donc six nœuds.



Choix des cases
Ayant établi l'arbre du jeu, chaque nœud terminal reçoit une valeur : 1 pour la victoire des croix, 0 pour une partie nulle, et -1 pour la victoire des cercles. Nous pouvons alors remonter l'arbre et affecter une valeur à chaque nœud. Si nous prenons le nœud le plus à droite du premier niveau, nous trouvons la valeur -1, à partir des deux nœuds inférieurs de valeur 0 et -1. Du fait que cette valeur dépend des ronds (c'est à eux de jouer), on prendra la valeur minimale (-1). En remontant jusqu'au nœud en cours, on peut conclure que les croix doivent choisir la valeur maximale parmi les trois possibles.

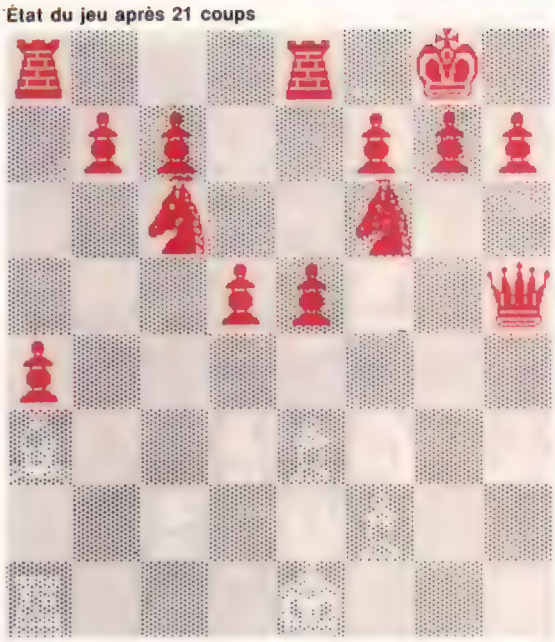
Liz Dixon

Mouvements

1	d2-d3	e7-e5
2	Cb1-d2	Cb8-c6
3	g2-g3	Cg8-f6
4	Ff1-g2	Ff8-c5
5	e2-e3	d7-d5
6	Cg1-e2	0-0
7	a2-a3	Fc8-f5
8	b2-b3	Cf6-g4?!
9	h2-h3	Cg4-f6
10	Fc1-b2	Dd8-d6
11	g3-g4	Ff5-e6
12	Ce2-g3	a7-a5
13	Dd1-e2	Cf6-d7
14	Cg3-f5	Fe6 x f5
15	g4 x f5	Cd7-f6
16	h3-h4	Tf8-e8
17	Fg2-h3	a5-a4
18	b3-b4	Fc5 x b4!?
19	a3 x b4	Dd6 x b4
20	Fb2-a3	Db4 x h4
21	Cd2-f3	Dh4-h5
22	Cf3-d2	Dh5 x e2+?
23	Re1 x e2	b7-b5
24	c2-c3	h7-h6
25	Fh3-g2	Ta8-a5
26	Ta1-b1	Te8-b8
27	Tb1-b2	Tb8-b6
28	Th1-b1	Rg8-h7
29	Fa3-c5	Tb6-b8
30	Fc5-a3	Cc6-a7
31	Cd2-f3	Cf6-d7
32	Cf3-e1	c7-c6
33	Ce1-c2	Tb8-a8
34	Cc2-b4	Ta8-d8
35	Cb4-a2	Ca7-c8
36	c3-c4!	d5 x c4
37	d3 x c4	b5 x c4
38	Fg2 x c6	Cc8-a7
39	Fc6-e4	Cd7-f6
40	Fa3-e7	Td8-c8
41	Fe7 x f6	g7 x f6
42	Tb2-b6!	c4-c3
43	Tb6 x f6	Rh7-g7
44	Tf6-b6	a4-a3
45	Tb1-g1+	

Le programme indiqua à ce point de la partie qu'il perdait par 2,4 pions. Ses programmeurs abandonnèrent alors en son nom.

État du jeu après 21 coups



Cette partie d'échecs provient d'un tournoi au cours duquel le programme le plus puissant, le Cray Blitz, fut battu 4-0 par David Levy de Intelligent Software. Cette confrontation avait fait l'objet d'un pari d'un montant de 5 000 $ entre M. Levy et les programmeurs du Blitz. Bien que le programme soit du niveau des meilleurs joueurs, le tournoi a montré qu'il reste beaucoup de chemin à faire aux ordinateurs avant de les pouvoir affronter.

Le même raisonnement peut être appliqué en remontant dans l'arbre depuis le nœud I. Parvenant à ce point, nous savons que G donne un score de 20. Le nœud H, avec 25 points, semble préférable, mais le joueur MINI (et non le joueur MAXI) choisit entre G et J, et préférera manifestement G. Il est donc inutile de chercher à savoir si I est plus intéressant, puisque MAXI ne pourra jamais y aller.

Nous pouvons mettre ces idées sous la forme d'un arbre généalogique. MAXI est un « macho » qui pense, par exemple, que le nœud C est l' « oncle » des nœuds D et E, tous les deux fils du même père. MINI, pour sa part, est une « féministe », et pense, pour ce qui la concerne, que G est la tante des sœurs H et I, dont la mère est J. Pour autant que vous n'êtes pas traumatisés par les changements de sexe des nœuds à chaque coup du jeu, cette analogie nous permet d'exposer de manière très concise la règle « alpha-bêta » :

- Lorsque MAXI trouve un fils *pire* que ses oncles, il ignore les autres frères de ce fils indigne.
- Lorsque MINI rencontre une fille *meilleure* que ses tantes, elle ignore les autres sœurs de cette fille prodige.

Au mieux, l'algorithme « alpha-bêta » ne calcule que deux fois la racine carrée du nombre affecté aux nœuds terminaux de l'arbre de jeu, par rapport à la méthode simple dite « minimaxi ». Au pire, il en examine autant, et plus lentement. Pour éviter le premier des deux cas donnés plus haut, il est essentiel de générer de manière cohérente les frères et sœurs à chaque niveau. Aux niveaux maximaux, « les plus mauvais d'abord » (du point de vue de l'adversaire : « les meilleurs d'abord »).

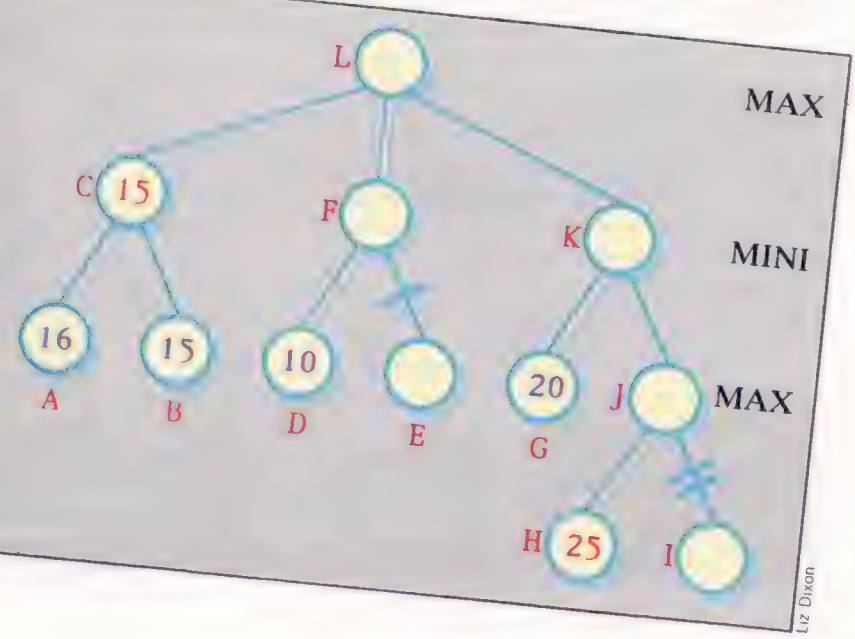
Pour illustrer les importants concepts de recherche par arbres, nous vous proposons un jeu d'intelligence artificielle qui ne fait intervenir que des techniques d'exploration des possibles. Cela signifie que les détails, tels que représentation de l'échiquier, règles de déplacement des pièces et évaluation statique (essentielles pour tout programme de jeu mais spécifiques à chacun d'eux), ne viennent pas entraver le déroulement de la procédure « alpha-bêta ».

Il n'y a ni échiquier, ni pièces. L'état du jeu est entièrement décrit en terme d'un seul nombre, noté V%. Le but du jeu pour l'ordinateur est d'atteindre la valeur 255 pour V% ; au joueur de faire en sorte qu'elle soit inférieure à 255.

Lorsque c'est son tour, le joueur choisit une des quatre fonctions disponibles (de A à D), listées aux lignes 1030-1060. Vous pouvez les modifier pour créer différentes versions, pour rendre

Élagage

La méthode d'élagage Alpha-Bêta améliore l'algorithme de base « minimalisation/maximalisation » en supprimant les embranchements redondants. Les suppressions de type Alpha (indiqués par un seul trait) interviennent lorsqu'un chemin minimal a déjà été trouvé à partir d'un déplacement du joueur MINI, et que d'autres recherches sont devenues superflues. Inversement, les élagages Bêta (un double trait), ont lieu lorsqu'un chemin maximal a été trouvé à la suite d'un déplacement du joueur MAX.



Liz Dixon

le jeu plus difficile à l'ordinateur par exemple. Ce jeu est très simple, mais il illustre bien la stratégie de recherche de solutions en intelligence artificielle en l'absence de détails superflus. Il est en outre très mathématique, ce qui donne à l'ordinateur un avantage naturel. L'algorithme optimisation/minimalisation « alpha-bêta » dépend largement, dans cette version, de l'utilisation de fonctions récursives avec paramètres et variables locales. Les paramètres sont les suivants :

VV% : état courant du jeu.
A% : Alpha-test à un niveau donné (meilleure valeur).
B% : Bêta-test à un niveau donné (plus mauvaise valeur).
D% : indicateur de niveau généalogique.

Ce programme suppose la récursion. Nous donnerons prochainement une version non récursive.

Le jeu des nombres

```
 10 REM ****************************
 11 REM **    Listage 3.1 :       **
 12 REM **  LE JEU DES NOMBRES    **
 13 REM ****************************
 50 MODE 7
100 REM -- Jeu illustrant la recherche:
120 GOSUB 1000  : REM initialisation
130 GOSUB 1500  : REM instructions
150 REPEAT
160   GOSUB 2000  : REM préparation nouveau jeu
170   INPUT "qui commence (1=Vous, 2=moi) ", H1%
180   IF H1%<1 OR H1%>2 THEN GOTO 170
200   REM -- boucle principale :
210   REPEAT
220     IF H1%=1 THEN GOSUB 3000
230     REM -- coup de l'utilisateur.
240     GOSUB 3500  : REM affichage de l'échiquier
250     H1%=1 : REM toujours 1 après le 1er cycle
260     GOSUB 4000  : REM test vainqueur
270     IF EG%=0 THEN GOSUB 5000
280     REM -- coup de l'ordinateur
290     GOSUB 3500  : REM affichage de l'état du jeu
300     GOSUB 4000  : REM test fin de partie
310     UNTIL EG%<>0 OR M%>33
320   REM -- fin :
330   GOSUB 6000 : REM bravo!
340   PRINT "une autre partie (N = NON) ";
350   Y$=GET$
360   UNTIL Y$="N" OR Y$="n"
365 PRINT
370 PRINT "A bientôt, et merci pour le jeu !"
400 END
444 :
500 DEF FNdéplacementmax(VV%,A%,B%,D%)
505 REM ---------- à moi :
510 LOCAL P%,E%,KEEP%
515 IF D%>=MD% OR ABS(VV%)>HI% THEN =VV%
    REM valeur statique
520 REM sinon aller plus loin :
525 P%=0
530 REPEAT  P%=P%+1
535   M%=P%: V%=VV%: GOSUB 5500: REM à vous de jouer
536   IF D%=1 THEN PRINT CHR$(M%+64);" = ";
540   E%=FNdéplacementmin(V%,A%,B%,D%+1)
545   IF E%>A% THEN A%=E%: KEEP%=P%
548   IF D%=1 THEN PRINT E%;" ";
550   UNTIL P%>3 OR A%>=B%
555 IF A% BV% AND D%=1 THEN BV%=A%: HH%=KEEP%
556 REM le meilleur jusqu'ici.
560 =A%
565 REM résultat avec max A%
570 :
700 DEF FNdéplacementmin(VV%,A%,B%,D%)
710 REM ---------- au tour de l'autre :
720 LOCAL E%,P%
730 IF D%>=MD% OR ABS(VV%)>HI% THEN =VV%
740 P%=0
750 REPEAT P%=P%+1: M%=P%
755   V%=VV%: GOSUB 5500: REM A vous de jouer
760   E%=FNdéplacementmax(V%,A%,B%,D%+1)
770   IF E%<B% THEN B%=E%
780   UNTIL P%>3 OR B%<=A%
790 =B%
796 REM résultats avec la plus petite valeur B%
999 :
1000 REM -- programme d'initialisation :
1001 BL$=""
1002 @%=4
1020 REM -- les 4 fonctions :
1030 DEF FNA(X%)=2*X% - 7
1040 DEF FNB(X%)=X  DIV 2 + 1
1050 DEF FNC(X%)=-4*X% + 17
1060 DEF FND(X%)=3*X% - 4
1070 LO%=-255: HI%=255
1150 RETURN
1160
1500 REM -- programme mode d'emploi :
1610 CLS: PRINT
1620 PRINT "Bienvenue au Jeu des Nombres !"
1630 PRINT "Si vous n'en connaissez pas les règles,"
1635 PRINT "LISEZ CECI !:"
1636 PRINT "N.B. Je maximalise : vous minimalisez."
1640 PRINT "pour voir l'effet d'un déplacement, tapez:"
1645 PRINT "A, B, C, oUD. "pour le valider, tapez R."
1650 PRINT : PRINT "Bonne chance !";CHR$(7)
1660 RETURN
1670
2000 REM -- Programme de préparation:
2010 M%=0 : REM déplacements
2020 V% = RND(15)-8 : REM état initial.
2050 EG%=0
2060 PRINT'"Etat initial = ";V%
2100 RETURN
2110
3000 REM -- Déplacement Utilisateur
3010 M%=M%+1
3020 PRINT
3030 REPEAT
3040   PRINT "Que jouez vous ? ";
3050   H$=GET$: PRINT H$;
3060   IF H$="A" THEN PRINT FNA(V%): M%=1
3070   IF H$="B" THEN PRINT FNB(V%): M%=2
3080   IF H$="C" THEN PRINT FNC(V%): M%=3
3090   IF H$="D" THEN PRINT FND(V%): M%=4
3100   UNTIL H$="X"
3120 GOSUB 5500 : REM Prenez une valeur pour
3150 RETURN
3160 :
3500 REM -- Programme d'affichage de l'échiquier :
3520 CLS : PRINT
3522 PRINT "    Déplacement ";M%;" --   ";
3523 IF M% 1 THEN RETURN
3525 PRINT CHR$(64+M%);
3530 PRINT " = ";V%
3535 RETURN
3700
4000 REM -- Programme-test de victoire (selon M%):
4001 IF M%<1 THEN RETURN
4010 EG%=0
4020 IF V%<LO% THEN EG%=-1
4030 IF V%>HI% THEN EG%=1
4040 RETURN
4050 :
5000 REM -- Programme de jeu de l'ordinateur :
5005 W%=V% : REM Sauvegarder l'état en cours.
5010 M%=M%+1
5015 MD%=6 : REM niveau max
5020 IF M%<4 THEN MD%=4
5030 IF M%>8 THEN MD%=8
5040 GOSUB 5200 : REM -->   M%
5045 V%=W% : REM restorer l'état
5050 GOSUB 5500 : REM faites-le
5070 RETURN
5080 :
5200 REM -- choix de déplacement :
5210 BV%=LO% : D%=0
5220 BV%=FNdéplacementmax (V%,LO%,HI%,1)
5230 M%=HH%
5240 PRINT'"Appuyez sur une touche pour continuer ";
5244 C%=GET
5250 RETURN
5270 :
5500 REM -- Programme de choix de déplacement (M% : V% ):
5505 ON M% GOTO 5510,5520,5530,5540
5510 V%=FNA(V%): RETURN
5520 V%=FNB(V%): RETURN
5530 V%=FNC(V%): RETURN
5540 V%=FND(V%): RETURN
5550 :
6000 REM -- Programme de félicitations :
6010 PRINT "FIN DE PARTIE !"
6020 IF EG%>0 THEN PRINT " J'ai gagné !! "
6030 IF EG%<0 THEN PRINT " Bravo ! "
6040 IF EG%=0 THEN PRINT " Match nul "
6050 RETURN
6600 :
```

Le prologue

PROLOG est un langage à la fois très utile et convivial. Nous allons, au cours d'une série d'articles sur PROLOG, définir sa logique de base et retracer son histoire.

Le temps où les Japonais affirmaient révolutionner la technologie informatique en investissant massivement dans des recherches sur la cinquième génération semble déjà bien loin. Une de leurs décisions les plus significatives fut peut-être d'adopter le langage de programmation, peu connu alors, PROLOG, comme noyau de base pour les bases de données intelligentes à hautes performances prospectives.

PROLOG est un acronyme pour « PROgrammation LOGique ». C'est, du reste, une bonne réalisation de cet idéal, même si elle est incomplète. Mais quel est l'intérêt d'une programmation logique? De nombreuses logiques peuvent être utilisées pour décrire le monde et ses divers aspects. Certaines nous sont familières, telles que les mathématiques. D'autres relèvent de théories ésotériques, voire de doctrines philosophiques. Le calcul de *prédicats* de premier ordre constitue une forme de logique proche de celle utilisée par la pensée au cours de la vie de tous les jours et lors de nos conversations habituelles. Elle n'en a pas moins sa propre notation et ses propres restrictions. Un prédicat doit être considéré comme une relation entre objets logiques. Dans la phrase « Jean aime Anne », le prédicat est « aime ». Cette relation pourrait alors s'écrire « aime (Jean, Anne) », ce qui est moins lisible mais davantage explicite quant à la distinction prédicat/arguments. Pour indiquer que Jean est un homme, nous pourrions écrire « homme(Jean) », « homme » étant un prédicat avec pour argument « Jean ». De la même manière, « femme(Anne) » signifie qu'Anne est une femme.

Il ne s'agit ici que de simples déclarations, mais nous pouvons étendre cette logique pour signifier que certains faits en impliquent d'autres. En utilisant la logique des prédicats, nous pouvons décrire le monde en termes de faits et d'implications. Nous utilisons alors les descriptions pour déduire de nouveaux faits à partir de faits connus. Pour cela, nous introduisons des variables. Les variables logiques ressemblent beaucoup à celles que vous utilisez en BASIC ou dans d'autres langages de programmation, si ce n'est que leur étendue est limitée à la clause (le fait ou l'implication) dans laquelle elles figurent, et non plus à l'ensemble des clauses. Cela signifie que le « Jean-qui-aime-Anne » peut ne pas être le même que le Jean qualifié « homme ».

Nous pourrions écrire un fait tel que « femme(X)→aime (Jean, X) ». La flèche signifie l'implication. Aussi pouvons-nous lire cette règle énonçant que « le fait que X soit une femme implique que Jean aime X ». En langage courant, cela se dit « Jean aime X si X est une femme ». Cette règle est un exemple de clause particulière dans le calcul de prédicat, appelée *clause d'en-tête*. Ces dernières comportent une déclaration préliminaire (la conséquence), qui n'est vraie que lorsque toutes les déclarations du corps de la clause sont vérifiées.

A si B et C si D

est une clause d'en-tête, avec A pour en-tête et B, C et D pour corpus (les antécédents). Il est possible de déclarer un simple fait sans antécédents; c'est alors une conséquence supposée implicitement vraie.

La logique pure nous permet d'écrire un programme comme un ensemble de faits et de règles décrivant les éléments qui nous intéressent. Cette description est dans sa forme très semblable à la manière dont nous concevons le problème. L'exécution de notre programme logique suppose de prouver la véracité ou la fausseté d'une affirmation. S'il s'agit d'un simple fait, nous pouvons affirmer d'emblée sa véracité. Si la déclaration est la conséquence d'une règle, alors il nous faut prouver la véracité de tous ses antécédents avant de pouvoir dire qu'elle est elle-même vraie. Ainsi, si nous voulons prouver que Jean est un homme (!), nous devons tester l'assertion « homme(Jean) », que nous savons être vraie puisque c'est un fait d'ori-

Prélude aux diverses versions de prolog

PROLOG nécessite généralement beaucoup de RAM pour une bonne mise en place. Les versions exécutables sur micro-ordinateur domestique sont rares. Les utilisateurs de Spectrum peuvent cependant acquérir MICRO-PROLOG, écrit par Logic Programming Associates et distribué par Sinclair sur cassette.

Les utilisateurs d'autres micros peuvent espérer pour bientôt une version spécifique, les éditeurs de logiciels s'intéressant de plus en plus à PROLOG.

MICRO-PROLOG de Spectrum diffère en de nombreux points du PROLOG DEC10 standard, la version utilisée dans nos exemples tout au long de cette suite d'articles. Nous tiendrons cependant une chronique des variantes de PROLOG pour le Spectrum, de sorte que ces listages pourront être directement saisis par MICRO-PROLOG.

Les principales différences seront exposées ultérieurement.

(Cl. Liz Heaney.)

gine. Mais si nous voulons savoir si « Jean aime Anne », nous devons d'abord prouver les déclarations « homme (Jean) » et « aime(Jean, Anne) ». PROLOG fut développé à l'université de Marseille au début des années soixante-dix par A. Colmerauer. Il utilise seulement les clauses d'en-tête de la logique des prédicats, et une notation qui ressemble à celle que nous avons indiquée. L'ensemble de la théorie de calcul des prédicats aurait pu être mis en œuvre, mais PROLOG, comme tous les langages de programmation, est un compromis entre efficacité informatique maximale et puissance d'expression. De nombreuses recherches ont actuellement lieu pour affiner cet équilibre; mais PROLOG sous sa forme actuelle semble assez abouti.

La version la plus proche du PROLOG standard est le DEC-10 PROLOG (du fait de sa première introduction sur un système de Digital Equipment Corporation). Un livre, dont le titre est « La programmation avec PROLOG », décrit les caractéristiques standards du langage et donne de nombreux détails pratiques. La plupart des versions courantes du langage PROLOG, dont le C-PROLOG, sont fondées sur ce standard, même s'il existe de nombreuses nuances et langages particuliers. Il existe deux versions essentielles pour les micro-ordinateurs : celle d'Expert Systems, très proche du standard, et MICRO-PROLOG de Logic Programming Associates, qui, bien que très répandue, diffère considérablement par sa syntaxe et sa structure interne.

Les mises en œuvre de PROLOG utilisent beaucoup de place mémoire. Aussi ne concernent-ils vraiment que les micros d'au moins 64 K de RAM.

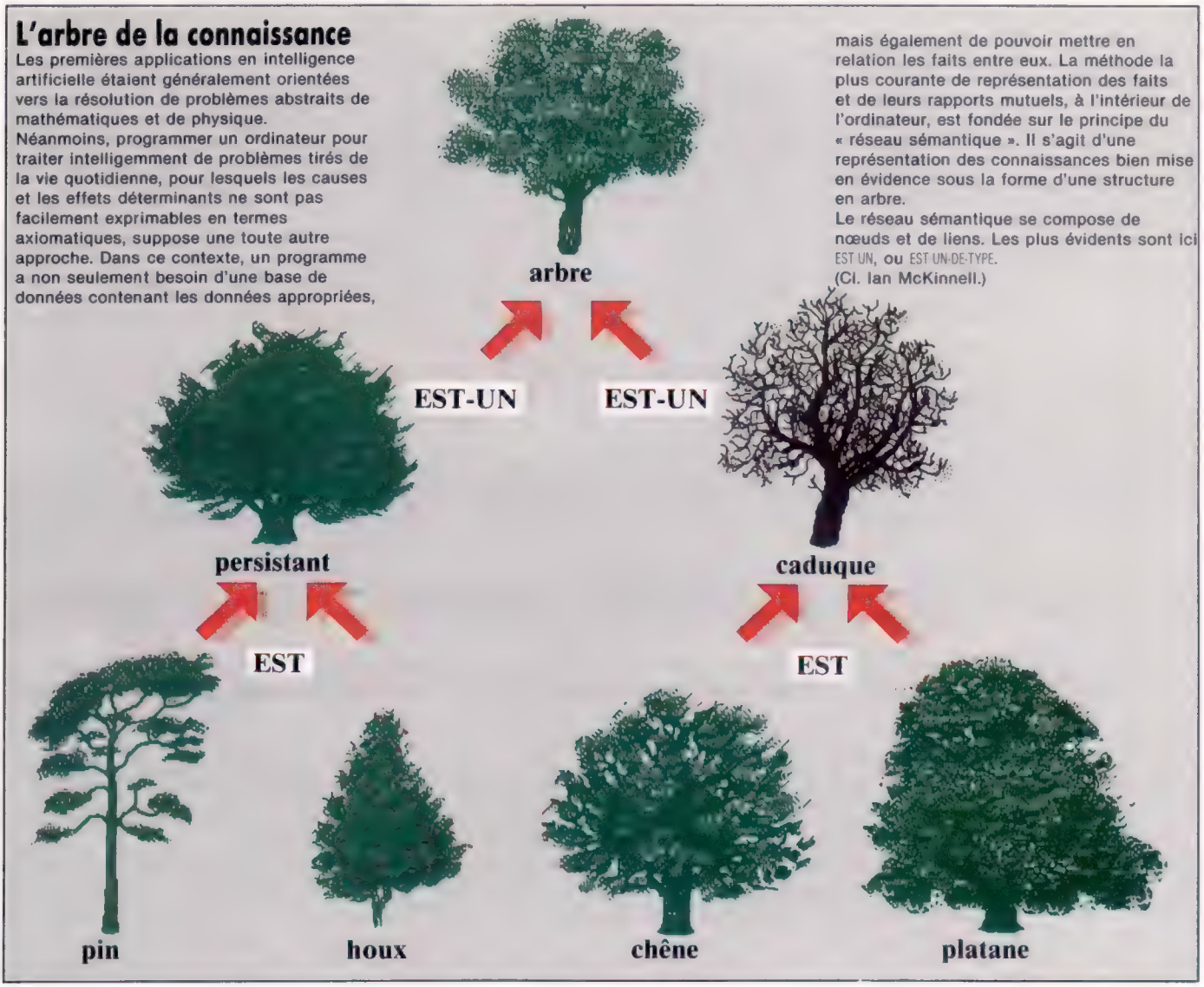
Les programmes écrits avec PROLOG ne suivent pas le déroulement habituel, dans lequel la première instruction est exécutée, puis la seconde, et ainsi de suite jusqu'à la dernière. Ils n'ont pas la structure traditionnelle des autres langages avec branchements et boucles. PROLOG utilise une méthode dite « remonter à la source ».

Pour résoudre le problème posé par une recherche, PROLOG progresse selon un enchaînement de règles se donnant à chaque fois un but nouveau à vérifier. Lorsqu'un embranchement se révèle infructueux, PROLOG remonte à un niveau antérieur pour prendre une nouvelle direction.

Cette démarche donne à PROLOG une personnalité très originale par rapport aux autres langages de programmation. Les partisans de la programmation logique insistent toujours sur la nature déclarative d'un programme PROLOG. Cela signifie lire une règle comme s'il s'agissait d'une clause d'un calcul de prédicat de premier ordre. Par exemple, X est l'oncle de Y si X est un homme, et X est parent de Z, et Z est le parent de Y. PROLOG peut néanmoins toujours être lu dans un style plus courant.

L'arbre de la connaissance

Les premières applications en intelligence artificielle étaient généralement orientées vers la résolution de problèmes abstraits de mathématiques et de physique.
Néanmoins, programmer un ordinateur pour traiter intelligemment de problèmes tirés de la vie quotidienne, pour lesquels les causes et les effets déterminants ne sont pas facilement exprimables en termes axiomatiques, suppose une toute autre approche. Dans ce contexte, un programme a non seulement besoin d'une base de données contenant les données appropriées, mais également de pouvoir mettre en relation les faits entre eux. La méthode la plus courante de représentation des faits et de leurs rapports mutuels, à l'intérieur de l'ordinateur, est fondée sur le principe du « réseau sémantique ». Il s'agit d'une représentation des connaissances bien mise en évidence sous la forme d'une structure en arbre.
Le réseau sémantique se compose de nœuds et de liens. Les plus évidents sont ici EST UN, ou EST-UN-DE-TYPE.
(Cl. Ian McKinnell.)



Révolution graphique

Cet article sur les graphiques 3-D (trois dimensions) sur Commodore 64 complète la conversion du listage BASIC et introduit des routines interpréteur en virgule flottante.

Le programme hybride (Test I-Rot et I-Rot.Hex) développé précédemment est assez vite exécuté. Néanmoins, il apparaît que le fait de parcourir la matrice d'adjacence E%(I,J) (qui définit les nœuds connectés dans la figure fil de fer) pour découvrir les points à tracer ralentit le programme. Pour accélérer les choses, il faut donc coder le reste de la boucle BASIC du programme Rotation de Cube en langage machine et exécuter l'ensemble de la boucle en langage machine. Le résultat est un gain de temps satisfaisant.

Afin d'exécuter les calculs suivants :

```
X1% = X(I) + 159:Y1% = 199 - (Z(I) + 100)
X2% = X(J) + 159:Y2% = 199 - (Z(J) + 100)
```

trouvés aux lignes 1640 et 1650 du programme BASIC initial, nous aurons besoin d'autres appels interpréteur. En fait, ces deux lignes de BASIC prennent une variable virgule flottante — soit X(I) — et additionnent 159 (en format virgule flottante) avant de prendre la partie entière et de la stocker en format deux octets comme X1%.

Les appels interpréteur nécessaires pour accomplir cela sont les suivants :

• FLPINT (adresse d'appel $B1AA) :
Cette routine prend la partie entière du nombre dans FAC et redonne le résultat (s'il est compris entre −32767 et 32767) dans le format Lo/Hi, respectivement dans les registres Y et A. Notez ici l'ordre inhabituel Lo/Hi, à l'inverse de la plupart des routines interpréteur.

• SNGFT (adresse d'appel $B3A2) :
Cette routine prend un entier à un octet (compris entre 0 et 255) du registre Y et le place dans FAC en format virgule flottante. SNGFT est utilisé dans la routine SETUP du listage d'assemblage (ligne 5150). Par exemple, la valeur décimale 159 est placée dans le registre d'index Y, et SNGFT est appelé pour la convertir et placer le résultat dans FAC. Après quoi, MOVMF est utilisé pour placer le résultat dans les cinq octets de MEM1. Ainsi, lorsqu'on veut additionner 159, il est disponible dans MEM1.

Les problèmes principaux qui restent dans la conversion de la boucle BASIC en langage machine concernent la spécification d'éléments particuliers dans les tableaux définissant la forme à faire tourner. Le calcul des pointeurs de tableau peut s'avérer difficile dans certains cas. Les tableaux de coordonnées X(I), Y(I), Z(I) ne présentent pas de difficultés spéciales, puisque dans chaque tableau nous ne faisons qu'ajouter cinq octets au pointeur pour obtenir l'adresse de l'élément suivant.

Le tableau E%(I,J), cependant, étant à deux dimensions, est entièrement différent. Voici son arrangement en mémoire :

```
E%(0,0)E%(1,0)E%(2,0)...E%(NP,0)
E%(0,1)E%(1,1)E%(2,1)...E%(NP,1), etc.
```

Autrement dit, le tableau consiste en blocs de mémoire de $2 \times (NP + 1)$ octets de longueur chacun — chaque bloc correspondant aux valeurs du second indice, et chaque élément prenant deux octets (puisqu'il s'agit d'un tableau entier).

Notre projet consiste à transcrire le code BASIC le plus fidèlement possible en langage machine; les boucles I, J (qui décrivent E%(I,J)) sont :

```
FORI=1 TONP
FORJ=1 TOI
```

Cela complique le changement dans le pointeur qui effectue l'équivalent en langage machine de NEXT I. Pour effectuer la rotation, il faut avoir accès aux éléments de E%(I,J) dans cet ordre :

```
E%(1,1)
E%(1,2)E%(2,2)
E%(1,3)E%(2,3)E%(3,3)
E%(1,4)E%(2,4)E%(3,4)E%(4,4) etc.
```

Un calcul rapide montre qu'il faut additionner $2 \times (NP + 1)$ au pointeur à chaque fois que I est incrémenté. Le meilleur moyen de transposer cela en langage machine 6502 est d'utiliser l'adressage indirect pour accéder à E%(I,J). Le code devient :

```
LDY JINDEX
LDA (ZPTEMP),Y
```

où ZPTEMP est un pointeur page zéro à deux octets, et JINDEX est utilisé pour garder trace de J. ZPTEMP doit aussi être incrémenté à chaque incrément de Z. En augmentant ZPTEMP et le registre Y, le décalé net est accru des deux octets nécessaires pour chaque incrément de J. Le résultat final de ces considérations est que ZPTEMP doit être incrémenté de : $(2 \times NP + 1)-(I-1)$ à chaque itération de la boucle I. (I-1) est soustrait de la longueur du bloc parce que ZPTEMP a déjà été incrémenté (I-1) fois dans la boucle (juste terminée) J. Utiliser cette expression pour calculer le décalé implique que ZPTEMP pointe l'octet correct après incrémentation de I.

Enfin, il serait commode de pouvoir appeler une routine interpréteur pour localiser la variable E%(I,J). Une telle routine existe, mais malheureusement elle est tortueuse et extrêmement lente, de sorte que nous devons faire le calcul du décalé à partir de l'adresse de E%(1,1).

Routines de rotation

Les listages suivants donnent le code source pour un programme en langage machine qui fait tourner une figure fil de fer définie par le tableau Elem.Jn, sur l'écran haute résolution. Le programme utilise les routines interpréteur et les méthodes de localisation des variables BASIC dont il est question dans le texte principal. Notre objectif de conversion de boucle BASIC parcourant les tableaux pour produire la figure en rotation est maintenant atteint. On donne aussi un chargeur BASIC pour ce code source, ainsi qu'un programme Test BASIC qui fixe les variables en langage machine et appelle la routine. Le programme test peut être utilisé avec une version assemblée du code source, II-Rot.Hex, ou en chargeant et en exécutant le programme chargeur, en tapant NEW et, enfin, le programme Test.

Chargeur de basic

```
1000 REM**  INSERT MSC II-ROT.HEX   **
1010 REM*******************************
1020 DATA72,138,72,152,72,32,181,199
1030 DATA173,83,197,172,84,197,32,162
1040 DATA187,173,75,197,172,76,197,32
1050 DATA40,186,162,94,160,197,32,212
1060 DATA187,173,87,197,172,88,197,32
1070 DATA162,187,173,77,197,172,78,197
1080 DATA32,40,186,169,94,160,197,32,90
1090 DATA184,162,94,160,197,32,212,187
1100 DATA173,87,197,172,88,197,32,162
1110 DATA187,173,75,197,172,76,197,32
1120 DATA40,186,162,99,160,197,32,212
1130 DATA187,173,83,197,172,84,197,32
1140 DATA162,187,173,77,197,172,78,197
1150 DATA32,40,186,169,99,160,197,32
1160 DATA183,184,162,99,160,197,32,212
1170 DATA187,169,94,160,197,32,162,187
1180 DATA174,83,197,172,84,197,32,212
1190 DATA187,169,99,160,197,32,162,187
1200 DATA174,87,197,172,88,197,32,212
1210 DATA187,206,83,197,240,31,169,5,24
1220 DATA109,83,197,141,83,197,144,3
1230 DATA238,84,197,169,5,24,109,87,197
1240 DATA141,87,197,144,3,238,88,197,76
1250 DATA112,197,169,1,141,0,193,141,1
1260 DATA193,141,2,193,32,14,193,32,101
1270 DATA199,169,1,141,81,197,141,83
1280 DATA197,173,79,197,133,253,173,80
1290 DATA197,133,254,172,82,197,177,253
1300 DATA200,3,76,207,198,173,83,197
1310 DATA72,84,197,32,162,187,169,94
1320 DATA160,197,32,103,184,32,170,177
1330 DATA140,0,195,141,1,195,173,85,197
1340 DATA172,84,197,32,162,187,169,94
1350 DATA160,197,32,103,184,32,170,177
1360 DATA140,2,195,141,3,195,173,89,197
1370 DATA172,88,197,32,162,187,169,99
1380 DATA160,197,32,80,184,32,170,177
1390 DATA140,4,195,173,91,197,172,92
1400 DATA197,32,162,187,169,99,160,197
1410 DATA32,80,184,32,170,177,140,5,195
1420 DATA173,0,195,205,2,195,208,19,173
1430 DATA1,195,205,3,195,208,11,173,4
1440 DATA195,205,5,195,208,3,76,207,198
1450 DATA32,14,195,173,81,197,205,82
1460 DATA197,240,40,169,5,24,109,85,197
1470 DATA141,85,197,144,3,238,86,197
1480 DATA169,5,24,109,91,197,141,91,197
1490 DATA144,3,238,92,197,230,253,208,2
1500 DATA230,254,238,82,197,76,73,198
1510 DATA173,74,197,205,81,197,240,88
1520 DATA169,5,24,109,83,197,141,83,197
1530 DATA144,3,238,84,197,169,5,24,109
1540 DATA89,197,141,89,197,144,3,238,90
1550 DATA197,169,1,24,109,74,197,10,56
1560 DATA237,81,197,24,105,1,24,101,253
1570 DATA133,253,165,254,105,0,133,254
1580 DATA173,68,197,141,95,197,173,69
1590 DATA197,141,86,197,173,70,197,141
1600 DATA91,197,173,73,197,141,92,197
1610 DATA169,1,141,82,197,238,81,197,76
1620 DATA73,198,104,168,104,170,104,96
1630 DATA173,68,197,141,83,197,141,85
1640 DATA197,173,69,197,141,84,197,141
1650 DATA86,197,173,70,197,141,87,197
1660 DATA173,71,197,141,88,197,173,74
1670 DATA197,141,93,197,173,72,197,141
1680 DATA89,197,141,91,197,173,73,197
1690 DATA141,90,197,141,92,197,160,159
1700 DATA32,162,179,162,94,160,197,32
1710 DATA212,187,160,99,32,162,179,162
1720 DATA99,160,197,32,212,187,96
1730 DATA79160:REM*CHECKSUM*
1740 FORI=50536TO51123
1750 READX:POKEI,X:CC=CC+X
1760 NEXT
1770 READX:IFX<>CCTHENPRINT"CHECKSUM ERROR":STOP
1780 PRINT"II-ROT.HEX INSTALLED OK"
```

Listage d'assemblage

```
1010 ;++++++++++++++++++++++++++++
1020 ;++      ROTSUB 64          ++
1030 ;++                         ++
1040 ;++++++++++++++++++++++++++++
1140 ;
1150 *       = $C544
1190 ; VARIABLES CALLED FROM BASIC
1200 ;
1210 XBASLO  *=*+1  ; POKE50500,X(I)LO
1220 XBASHI  *=*+1  ; POKE50501,X(1)HI
1230 YBASLO  *=*+1  ; POKE50502,Y(1)LO
1240 YBASHI  *=*+1  ; POKE50503,Y(1)HI
1250 ZBASLO  *=*+1  ; POKE50504,Z(1)LO
1260 ZBASHI  *=*+1  ; POKE50505,Z(1)HI
1270 NP      *=*+1  ; POKE50506,NP
1280 CSLO    *=*+1  ; POKE50507,CSLO
1290 CSHI    *=*+1  ; POKE50508,CSHI
1300 SNLO    *=*+1  ; POKE50509,SNLO
1310 SNHI    *=*+1  ; POKE50510,SNHI
1320 EBASLO  *=*+1  ; POKE50511,EX(1,1)LO
1330 EBASHI  *=*+1  ; POKE50512,EX(1,1)HI
1340 ;
1350 ; VARIABLES USED BY M/C
1370 IINDEX  *=*+1
1380 JINDEX  *=*+1
1390 XILO    *=*+1
1400 XIHI    *=*+1
1410 XJLO    *=*+1
1420 XJHI    *=*+1
1430 YILO    *=*+1
1440 YIHI    *=*+1
1450 ZILO    *=*+1
1460 ZIHI    *=*+1
1470 ZJLO    *=*+1
1480 ZJHI    *=*+1
1490 TEMPNP  *=*+1
1500 MEM1    *=*+5 ;FLOATING POINT VAR
1510 MEM2    *=*+5 ;FLOATING POINT VAR
1520 ZPTEMP  =$FD  ;FREE ZERO PAGE LOC
1530 ;
1540 ;INTERPRETER ARITHMETIC CALLS
1550 FMULT   =$BA28 ; FAC=FAC*MEM
1570 FSUB    =$B850 ; FAC=MEM-FAC
1580 FADD    =$B867 ; FAC=FAC+MEM
1590 MOVFM   =$BBA2 ; FAC=MEM
1600 MOVMF   =$BBD4 ; MEM=FAC
1610 FLPINT  =$B1AA ; .Y/.A=INT(FAC) N.B. HI/LO ORDER !!
1620 SNGFT   =$B3A2 ; FAC= .Y
1630 ;
1640 ;OTHER MACHINE CODE ROUTINE/VARS
1660 LINSUB  =$C30E
1670 MLO     =$C300
1680 MHI     =$C301
1690 NLO     =$C302
1700 NHI     =$C303
1710 Y1      =$C304
1720 Y2      =$C305
1740 ;++++ SAVE REGISTERS++
1760         PHA
1770         TXA
1780         PHA
1790         TYA
1800         PHA
1820 ;++++ INITIALISE VARIABLES ++++
1840         JSR SETUP
1860 ;++++PERFORM  MEM1=X(I)*CS-Y(I)*SN
1880 START
1890         LDA XILO
1900         LDY XIHI
1910         JSR MOVFM  ; FAC = X(I)
1920         LDA CSLO
1930         LDY CSHI
1940         JSR FMULT  ; FAC = X(I)*CS
1950         LDX #<MEM1
1960         LDY #>MEM1
1970         JSR MOVMF  ; MEM1=X(I)*CS
1980         LDA YILO
1990         LDY YIHI
2000         JSR MOVFM  ; FAC = Y(I)
2010         LDA SNLO
2020         LDY SNHI
2030         JSR FMULT  ; FAC = Y(I)*SN
```

```
2040          LDA #<MEM1
2050          LDY #>MEM1
2060          JSR FSUB    ; FAC = MEM1-FAC
2070          LDX #<MEM1
2080          LDY #>MEM1
2090          JSR MOVMF   ; MEM1= FAC
2110 ;++++PERFORM MEM2=Y(I)*CS+X(I)*SN
2130          LDA YILO
2140          LDY YIHI
2150          JSR MOVFM   ; FAC = Y(I)
2160          LDA CSLO
2170          LDY CSHI
2180          JSR FMULT   ; FAC = Y(I)*CS
2190          LDX #<MEM2
2200          LDY #>MEM2
2210          JSR MOVMF   ; MEM2= Y(I)*CS
2220          LDA XILO
2230          LDY XIHI
2240          JSR MOVFM   ; FAC = X(I)
2250          LDA SNLO
2260          LDY SNHI
2270          JSR FMULT   ; FAC = X(I)*SN
2280          LDA #<MEM2
2290          LDY #>MEM2
2300          JSR FADD    ; FAC = MEM2+FAC
2310          LDX #<MEM2
2320          LDY #>MEM2
2330          JSR MOVMF   ; MEM2= FAC
2350 ;++++PERFORM X(I)=MEM1;Y(I)=MEM2
2370          LDA #<MEM1
2380          LDY #>MEM1
2390          JSR MOVFM   ; FAC = MEM1
2400          LDX XILO
2410          LDY XIHI
2420          JSR MOVMF   ; X(I)= FAC
2430          LDA #<MEM2
2440          LDY #>MEM2
2450          JSR MOVFM   ; FAC = MEM2
2460          LDX YILO
2470          LDY YIHI
2480          JSR MOVMF   ; Y(I)=FAC
2500 ;++++TEST END LOOP
2520          DEC TEMPNP
2530          BEQ CONTIN
2550 ;++++INCREMENT ARRAY POINTERS
2570          LDA #$05
2580          CLC
2590          ADC XILO
2600          STA XILO
2610          BCC XNOHI
2620          INC XIHI
2630 XNOHI
2640          LDA #$05
2650          CLC
2660          ADC YILO
2670          STA YILO
2680          BCC YNOHI
2690          INC YIHI
2700 YNOHI
2710          JMP START
2730 ;++++ CLEAR/INIT SCREEN
2750 CONTIN
2760          LDA #$01
2770          STA $C100
2780          STA $C101
2790          STA $C102
2800          JSR $C10E ; INIT HIRES
2820 ;++++ PLOT LINES FROM E%(I,J)
2840 ; INITIALISE VARIABLES
2860          JSR SETUP
2870          LDA #$01
2880          STA IINDEX     ; I=1
2890          STA JINDEX     ; J=1
2900          LDA EBASLO     ; STORE E%
2910          STA ZPTEMP     ; POINTER
2920          LDA EBASHI     ; ON ZERO
2930          STA ZPTEMP+1   ; PAGE
2950 ; OFF WE GO - START GIANT I/J LOOP
2970 NEXTIJ
2980          LDY JINDEX
2990          LDA (ZPTEMP),Y ;GET E%(I,J)
3000          BNE DOIT
3010          JMP ONWARD
3030 ;++++PERFORM  X1%=X(I)+159
3040 ;    MLO=X1% LO:MHI=X1% HI
3060 DOIT
3070          LDA XILO
3080          LDY XIHI
3090          JSR MOVFM       ; FAC = X(I)
3100          LDA #<MEM1
3110          LDY #>MEM1
3120          JSR FADD        ; FAC = X(I)+159
3130          JSR FLPINT      ; .Y/.A = INT(FAC)
3140          STY MLO         ; X1% LO
3150          STA MHI         ; X1% HI
3170 ;++++PERFORM  X2%=X(J)+159
3180 ;    NLO=X2% LO:NHI=X2% HI
3200          LDA XJLO
3210          LDY XJHI
3220          JSR MOVFM       ; FAC = X(J)
3230          LDA #<MEM1
3240          LDY #>MEM1
3250          JSR FADD        ; FAC = X(J)+159
3260          JSR FLPINT      ; .Y/.A = INT(FAC)
3270          STY NLO         ; X2% LO
3280          STA NHI         ; X2% HI
3300 ;++++PERFORM Y1%=99-Z(I)
3320          LDA ZILO
3330          LDY ZIHI
3340          JSR MOVFM       ; FAC =Z(I)
3350          LDA #<MEM2
3360          LDY #>MEM2
3370          JSR FSUB        ; FAC = 99-Z(I)
3380          JSR FLPINT      ; .Y/.A = INT(FAC)
3390          STY Y1          ; Y1%
3410 ;++++PERFORM Y2%=99-Z(J)
3430          LDA ZJLO
3440          LDY ZJHI
3450          JSR MOVFM       ; FAC =Z(J)
3460          LDA #<MEM2
3470          LDY #>MEM2
3480          JSR FSUB        ; FAC = 99-Z(J)
3490          JSR FLPINT      ; .Y/.A = INT(FAC)
3500          STY Y2          ; Y2%
3520 ; GET READY FOR LINSUB
3540 ;TWO BYTE COMPARISON FOR X1%=X2%
3560          LDA MLO
3570          CMP NLO
3580          BNE NOPE
3590          LDA MHI
3600          CMP NHI
3610          BNE NOPE
3630 ;NOW COMPARE Y1=Y2
3650          LDA Y1
3660          CMP Y2
3670          BNE NOPE
3690 ; ALL EQUAL SO AVOID LINESUB
3710          JMP ONWARD
3720 NOPE
3730          JSR LINSUB      ;PLOT LINE
3750 ; INCREMENT J POINTERS
3770 ONWARD
3780          LDA JINDEX      ;FIRST TEST
3790          CMP JINDEX      ;J=<I ?
3800          BEQ NEXTI
3820 ;INCREMENT XJLO/XJHI
3840          LDA #$05
3850          CLC
3860          ADC XJLO
3870          STA XJLO
3880          BCC XJNOHI
3890          INC XJHI
3900 XJNOHI
3920 ;INCREMENT ZJLO/ZJHI
3940          LDA #$05
3950          CLC
3960          ADC ZJLO
3970          STA ZJLO
3980          BCC ZJNOHI
3990          INC ZJHI
4000 ZJNOHI
4020 ;INCREMENT E% POINTER BY 1
4040          INC ZPTEMP
4050          BNE NOHIBY
4060          INC ZPTEMP+1
4070 NOHIBY
4080          INC JINDEX
4090          JMP NEXTIJ
4110 ; INCREMENT I POINTERS
4130 NEXTI
4140          LDA NP          ;FIRST TEST
4150          CMP IINDEX      ;I=<NP ?
4160          BEQ EXIT
4180 ; INCREMENT XILO/XIHI
4200          LDA #$05
4210          CLC
4220          ADC XILO
4230          STA XILO
4240          BCC XINOHI
4250          INC XIHI
4260 XINOHI
4280 ; INCREMENT ZILO/ZIHI
4300          LDA #$05
4310          CLC
4320          ADC ZILO
4330          STA ZILO
4340          BCC ZINOHI
4350          INC ZIHI
4360 ZINOHI
4380 ; INCREMENT E% POINTERS (TRICKY)
4400          LDA #$01        ; WHEN INC I
4410          CLC             ; MUST ADD
4420          ADC NP          ;2*(NP-1)-(I-1)
4430          ASL A           ; TO THE ZP
4440          SEC             ; POINTER TO
4450          SBC IINDEX      ; E%
4460          CLC
4470          ADC #$01
4480          CLC
4490          ADC ZPTEMP
4500          STA ZPTEMP
4510          LDA ZPTEMP+1
```

```
4520          ADC #$00
4530          STA ZPTEMP+1
4550 ; REINITIALISE XJLO/XJHI
4570          LDA XBASLO
4580          STA XJLO
4590          LDA XBASHI
4600          STA XJHI
4620 ; REINITIALISE ZJLO/ZJHI
4640          LDA ZBASLO
4650          STA ZJLO
4660          LDA ZBASHI
4670          STA ZJHI
4690 ; REINITIALISE JINDEX
4710          LDA #$01
4720          STA JINDEX
4740 ; INCREMENT IINDEX
4760          INC IINDEX
4770          JMP NEXTIJ
4790 ; GIANT LOOP ENDS HERE
4800 ;
4810 ;++++ PULL REGISTERS ++++
4820 ;
4830 EXIT
4840          PLA
4850          TAY
4860          PLA
4870          TAX
4880          PLA
4890          RTS
4930 ;++++ SUBROUTINE
4940 ;
4950 SETUP
4960          LDA XBASLO
4970          STA XILO
4980          STA XJLO
4990          LDA XBASHI
5000          STA XIHI
5010          STA XJHI
5020          LDA YBASLO
5030          STA YILO
5040          LDA YBASHI
5050          STA YIHI
5060          LDA NP
5070          STA TEMPNP
5080          LDA ZBASLO
5090          STA ZILO
5100          STA ZJLO
5110          LDA ZBASHI
5120          STA ZIHI
5130          STA ZJHI
5140          LDY #159
5150          JSR SNGFT     ; FAC = 159
5160          LDX #<MEM1
5170          LDY #>MEM1
5180          JSR MOVMF     ; MEM1=159 IN FPT
5190          LDY #99
5200          JSR SNGFT     ; FAC = 99
5210          LDX #<MEM2
5220          LDY #>MEM2
5230          JSR MOVMF     ; MEM2=99 IN FPT
5240          RTS
```

Programme test basic

```
1000 REM**PLOT ROTATING CUBES**
1010 IFA=0THENA=1:LOAD"PLOTSUB.HEX",8,1
1020 IFA=1THENA=2:LOAD"LINESUB.HEX",8,1
1030 IFA=2THENA=3:LOAD"II-ROT.HEX",8,1
1040 PRINTCHR$(147)"WAIT 17 SECS".
1050 REM**DIMENSION ARRAYS**
1060 NP=24:REM NUMBER OF POINTS
1070 DIM X(NP),Y(NP),Z(NP)
1080 DIM EX(NP,NP):REM EDGE CONNECTIONS
1090 REM**INITIALISE ARRAYS**
1100 REM--INITIAL CUBE COORDINATE DATA
1110 DATA  50, 50, 50:REM----------Y1
1120 DATA -50, 50, 50:REM TOP FOUR Y2
1130 DATA -50,-50, 50:REM POINTS   Y3
1140 DATA  50,-50, 50:REM----------Y4
1150 DATA  50, 50,-50:REM----------Y5
1160 DATA -50, 50,-50:REM BOT FOUR Y6
1170 DATA -50,-50,-50:REM POINTS   Y7
1180 DATA  50,-50,-50:REM----------Y8
```

La section suivante de programme lit les données définies ci-dessus pour remplir les tableaux X(), Y() et Z(). Notez que, pour les petits cubes, les valeurs données initiales sont réduites d'un facteur 0,3.

```
1190 REM**READ IN DATA FOR 3-CUBES**
1200 FORI=1TO8:REM CENTRE BIG CUBE
1210 READX(I),Y(I),Z(I)
1220 NEXT
1230 RESTORE
1240 FORI=9TO16:REM RIGHT SMALL CUBE
1250 READX,Y,Z
1260 X(I)=.3*X+120:Y(I)=.3*Y:Z(I)=.3*Z
1270 NEXT
1280 RESTORE
1290 FORI=17TO24:REM LEFT SMALL CUBE
1300 READX,Y,Z
1310 X(I)=.3*X-120:Y(I)=.3*Y:Z(I)=.3*Z
1320 NEXT
```

La section suivante fait tourner les coordonnées des points définissant les trois cubes pour obtenir une perspective à 45°.

```
1330 REM**ROTATE SPACE ABOUT X-AXIS ¶/4
1340 FORI=1TONP
1350 Y(I)=Y(I)*COS(¶/4)-Z(I)*SIN(¶/4)
1360 Z(I)=Z(I)*COS(¶/4)+Y(I)*SIN(¶/4)
1370 NEXT
1380 REM**ROTATE SPACE ABOUT Z-AXIS ¶/4
1390 FORI=1TONP
1400 X(I)=X(I)*COS(¶/4)-Y(I)*SIN(¶/4)
1410 Y(I)=Y(I)*COS(¶/4)+X(I)*SIN(¶/4)
1420 NEXT
```

La section suivante définit les points à connecter à la figure pour chacun des trois cubes utilisant le tableau d'adjacence :

```
1430 REM**EDGE CONNECTION DATA**
1440 REM-MIDDLE BIG CUBE-
1450 EX(1,2)=1:REM 1 CONNECTED TO 2
1460 EX(2,3)=1:EX(3,4)=1:EX(4,1)=1
1470 EX(5,6)=1:REM BOT SQUARE
1480 EX(6,7)=1:EX(7,8)=1:EX(8,5)=1
1490 EX(5,1)=1:REM TOP TO BOT EDGES
1500 EX(6,2)=1:EX(7,3)=1:EX(8,4)=1
1510 REM-----------------
1520 REM-RIGHT SMALL CUBE
1530 FORI=9TO16:FORJ=9TO16
1540 EX(I,J)=EX(I-8,J-8)
1550 NEXT:NEXT
1560 REM-LEFT SMALL CUBE
1570 FORI=17TO24:FORJ=17TO24
1580 EX(I,J)=EX(I-8,J-8)
1590 NEXT:NEXT
1600 REM**SYMMETRISE EX(I,J)**
1610 FORI=1TONP:FORJ=1TONP
1620 IFEX(I,J)<>0THENEX(J,I)=1
1630 NEXT:NEXT
```

Maintenant le programme appelle le langage machine pour effectuer les calculs de rotation nécessaires et tracer la nouvelle figure. Cet appel est fait à plusieurs reprises à partir de l'intérieur d'une boucle faisant tourner les trois cubes de 360° avant de terminer.

```
1640 REM**********************
1650 REM**PLOT ROTATING CUBE**
1660 SA=2¶/45:CS=COS(SA):SN=SIN(SA)
1670 GOSUB1790:REM INITIALISE
1680 FOR A=0 TO 2¶ STEP SA
1690 SYS50536:REM ROTATE
1700 NEXTA:REM NEXT ANGLE
1710 GETA$:IFA$=""THEN1710
1720 REM**********************
1730 GOSUB1750:REM RESET SCREEN
1740 END
1750 REM**RESET SCREEN**
1760 POKE49408,0:SYS49422
1770 PRINTCHR$(147)
1780 RETURN
1790 REM**INITIALISE ROTSUB**
1800 REM**********************
1810 REM**N.B. DO NOT DEFINE NEW**
1820 REM**VARIABLES BETWEEN THIS**
1830 REM** SUBROUTINE & CALLING **
1840 REM**  MACHINE CODE ROTATE **
1850 REM** AS THIS WOULD CHANGE **
1860 REM** BASE ADDRESS ARRAYS  **
1870 REM**********************
1880 X(1)=X(1):REM X(1) CURRENT VAR
1890 POKE50500,PEEK(71):REM X(1)LO
1900 POKE50501,PEEK(72):REM X(1)HI
1910 Y(1)=Y(1):REM Y(1) CURRENT VAR
1920 POKE50502,PEEK(71):REM Y(1)LO
1930 POKE50503,PEEK(72):REM Y(1)HI
1940 Z(1)=Z(1):REM Z(1) CURRENT VAR
1950 POKE50504,PEEK(71)
1960 POKE50505,PEEK(72)
1970 POKE50506,NP:REM NUMBER OF POINTS
1980 CS=CS:REM MAKE CS CURRENT VAR
1990 POKE50507,PEEK(71)
2000 POKE50508,PEEK(72)
2010 SN=SN:REM MAKE SN CURRENT VAR
2020 POKE50509,PEEK(71)
2030 POKE50510,PEEK(72)
2040 EX(1,1)=EX(1,1):REM EX CURRENT VAR
2050 POKE50511,PEEK(71)
2060 POKE50512,PEEK(72)
2070 RETURN
```

Les voix de la SNCF

Le secteur de la recherche de la SNCF s'intéresse depuis plusieurs années au traitement de la parole, et réalise actuellement des expériences de synthèse et de reconnaissance vocales.

Synthèse et reconnaissance
Les dispositifs de reconnaissance de la voix ont encore une mémoire limitée. Le vocabulaire des matériels actuellement disponibles représente moins de 200 mots. (Cl. SNCF.)

Le développement actuel de l'informatique à la Société nationale des chemins de fer français (SNCF) a entraîné la nécessité d'étudier sur tous les plans — technologique, économique, ergonomique — des outils de communication homme-machine.

Jusqu'à présent, ceux-ci se composaient d'entrées-sorties alphanumériques, sur clavier, écran, imprimante ou panneau d'affichage, et graphiques, sur table traçante ou écran. L'apparition d'entrées-sorties vocales est susceptible de modifier considérablement la situation de l'homme face à la machine, en raison de la nature même de la communication parlée.

En effet, la parole, en entrée d'ordinateur, permet de libérer l'opérateur de son poste de travail, lui laissant les mains libres et le regard disponible; il peut également se déplacer dans un certain périmètre. La parole est un moyen de communication plus rapide que la frappe d'une instruction au clavier, et surtout plus familier, plus naturel et plus spontané.

La voix de son maître

L'état actuel de la technique ne permet encore, cependant, que l'utilisation de dispositifs de reconnaissance monolocuteurs et dont la mémoire est limitée à un vocabulaire de quelque cent cinquante mots. Il existe donc des contraintes dans l'utilisation de ces dispositifs; en particulier, ceux-ci nécessitent une phase d'apprentissage où le locuteur « apprend » à l'ordinateur les mots qui devront être reconnus.

Divers systèmes existant sur le marché français et étranger ont été essayés en milieu ferroviaire. Une application immédiate réside dans la commande vocale à distance. Ainsi l'opérateur se trouvant sur le terrain pourra entrer les données en machine à l'aide d'un classique émetteur radio portatif qui expédiera les informations verbales jusqu'à l'ordinateur chargé de la reconnaissance.

Une autre application importante concerne la bureautique. A titre d'exemple, l'homme lira à haute voix les tableaux statistiques à traiter, et la machine, après avoir confirmé le message « entendu », lui répondra en signalant éventuellement quelque anomalie ou erreur.

Enfin, de simples ordres pourront être oralement lancés à destination d'une machine quelconque, qui les exécutera aussitôt. Le conducteur de train pourrait même avoir le loisir d' « interroger » oralement le tableau de bord. D'aucuns penseront peut-être que de tels équipements relèvent du gadget; pourtant, ils s'avèrent susceptibles, en bien des cas, d'améliorer la sécurité, voire surtout d'augmenter singulièrement la puissance de travail de l'homme, tout en le soulageant d'astreignantes manipulations. Au poste de commandement de Reims, les spécialistes du

secteur Recherche testent actuellement un dispositif utilisant la reconnaissance de la parole, spécialement mis au point en collaboration avec le Laboratoire central de télécommunications (L.C.T.) : le composeur d'appels téléphoniques à commande vocale.

Le régulateur de la cabine n° 3, assurant la commande centralisée de la voie unique Épernay-Reims et supervisant la circulation entre Dormans et Revigny sur la ligne Paris-Strasbourg, peut désormais entrer en communication sélective avec n'importe quelle gare ou poste de son secteur de régulation, en prononçant simplement leur nom. L'intervention vocale remplace ainsi l'intervention manuelle consistant à appuyer sur une touche d'un clavier.

Le matériel mis en œuvre à Reims inclut un mini-ordinateur à microprocesseur 16 bits, dissimulé derrière le meuble de régulation, un petit clavier extra-plat avec affichage à diodes électroluminescentes (DEL) posé sur la table de travail et servant au dialogue technique avec l'ordinateur, et enfin un simple microphone-cravate, bien moins contraignant qu'un micro-casque. Ce matériel est étrangement complété par... un lecteur de cassette. Car le dispositif de reconnaissance de la parole utilisé à Reims, étant « monolocuteur », ne reconnaît que « la voix de son maître » , préalablement enregistrée au cours d'une phase d'apprentissage.

Lors de cet apprentissage, l'opérateur — autrement dit, chaque régulateur, à sa prise de service — « charge » l'ordinateur en insérant dans le lecteur une cassette numérique qu'il lui fait « lire ». Chaque régulateur possède ainsi sa cassette où est sauvegardé le répertoire des « formes vocales » qui lui sont propres.

Après environ deux minutes, pendant lesquelles la cassette est lue et entrée en mémoire, le processus de « reconnaissance » se déclenche automatiquement, chaque fois que le régulateur prononce le nom d'une gare. L'ordinateur compare aussitôt la forme vocale émise à chacune des formes de référence du répertoire en mémoire. Lorsque cette forme se rapproche suffisamment de l'une des formes du répertoire, elle est considérée comme reconnue. Son nom apparaît en clair sur l'affichage DEL, et l'interface de sortie établit alors la communication téléphonique. Dans le cas contraire, l'ordinateur affiche la mention « veuillez répéter ».

Pour éviter que tous les noms de gares incidemment prononcés par le régulateur dans une conversation n'entraînent autant d'appels pour lesdites gares, une pédale actionnée par le régulateur provoque l'isolement du micro-cravate.

Assurément, le dispositif installé à Reims fonctionne déjà remarquablement bien. Le taux de reconnaissance dès le premier appel atteint 95 % et, si le mot est répété, on arrive à 99 % de réussite. Il s'agit, pour le moment, de mots isolés, mais, dès cette anneé (1985), les spécialistes du secteur de la recherche comptent expérimenter des appareils reconnaissant les mots « connectés » et, d'ici deux à trois ans, une troisième génération devrait permettre de comprendre la parole continue. Par ailleurs, les dispositifs de demain seront « multilocuteurs », c'est-à-dire reconnaîtront les propos de n'importe quel locuteur, sans nécessiter la phase d'apprentissage préalable.

La SNCF est actuellement le seul réseau européen à travailler sur la reconnaissance de la parole. Cette technique ne représente, en fait, que le premier aspect du traitement de la parole. L'aspect complémentaire, la « synthèse », déjà mieux maîtrisée, permet d'instaurer un véritable dialogue interactif entre l'homme et la machine.

Les divers procédés de synthèse

• *Parole numérisée :* la parole est modulée par impulsions codées. L'échantillonnage à 8 000 Hz (soit 8 000 échantillons par seconde) peut porter soit sur la totalité des mots (1 seconde de parole stockée dans 64 Kbits), soit sur des parties de mots, les « phones » (1,5 seconde de parole stockée dans 32 Kbits).
• *Synthèse par mots :* le synthétiseur restitue de la parole comprimée; ce n'est plus l'amplitude du signal qui est codée, mais seulement certaines caractéristiques fréquentielles et certains paramètres de la voix (1 seconde de parole stockée dans 2 400 bits à 600 bits).
• *Synthèse par éléments acoustiques :* ce ne sont plus les mots entiers, mais les éléments constitutifs du son (phonèmes ou diphonèmes) qui sont synthétisés, puis juxtaposés. (On admet, habituellement, dans la langue française, une trentaine de phonèmes — sons élémentaires — et un millier de diphonèmes — association de deux phonèmes.)

Une puce annonce les trains

« Montiéramey, voie 2, vingt-deux heures cinquante et une. » Cette voix immuable, retentissant dans la cabine de régulation n° 4 du P.C. de Reims, n'est pas, en dépit des apparences, celle d'un agent présent dans ladite gare. Elle est artificiellement élaborée, à quelques mètres du régulateur, par un ordinateur.

L'expérience de la synthèse de la parole a également commencé à Reims, avec l'expérimentation du premier « Trainphone ». Ce dispositif, faisant appel à la parole synthétique, renseigne le régulateur sur les heures de passage des trains dans les gares. Un appareil similaire peut s'adapter sur un système de suivi des circulations — « système d'annonce automatique des trains » (S.A.T.) — et, en outre, prononcer ainsi tous les numéros des trains.

De la sorte, le régulateur reste maître des informations vocales. Le « Trainphone » se compose du synthétiseur proprement dit, d'une horloge numérique et d'un amplificateur, ces trois derniers éléments étant installés derrière le meuble de régulation.

Par ailleurs, les annonces aux voyageurs paraissent d'ores et déjà constituer un domaine d'application privilégié de la synthèse de la parole. Au départ, il s'agissait d'une juxtaposition de messages préalablement enregistrés sur bande magnétique, mais ce procédé entraînait quelques problèmes de maintenance inhérents

Voix synthétiques
Des systèmes de saisie et de transmission automatique des prévisions de retard par télématique vocale sont en cours de mise au point. Ils diffusent automatiquement des informations par synthèse de paroles, via le réseau téléphonique SNCF, à des téléphones situés dans divers chantiers. C'est un micro-ordinateur qui pilote le synthétiseur et une mini-imprimante, et qui appelle les utilisateurs.
(Cl. Eric Garraud/SNCF.)

aux pièces tournantes des magnétophones; sans compter les modifications afférentes aux changements de service, requérant à chaque fois le concours de speakers professionnels... Avec les « puces parlantes », on obtient une fiabilité presque absolue, assortie d'une étonnante souplesse d'exploitation.

Ainsi, à la station « boulevard Victor », sur la « transversale rive gauche » (T.R.G.), l'ordinateur ne se contente plus d'afficher sur les panneaux les diverses informations concernant les trains. Désormais, il pilote également une unité vocale composant automatiquement des annonces du style : « Attention quai B, voie 3; la destination du train vient d'être modifiée; veuillez consulter le tableau d'affichage. » Y a-t-il un seul habitué de la gare du boulevard Victor ou des Invalides à s'être aperçu que la voix diffusée par les haut-parleurs n'était parfois plus d'origine humaine?

A Reims, encore, a eu lieu une expérimentation ayant pour cadre la gare de voyageurs. Chaque jour, durant quarante minutes, toutes les annonces intégrant l'ensemble des données habituelles (numéro du train, de la voie, estimation du retard éventuel, etc.) sont mémorisées ou composées par l'opérateur à l'aide d'un ensemble clavier-écran, pour être diffusées, soit en voix numérisée, soit en synthèse par mots (voir encadré).

Les premiers résultats observés à la SNCF sont considérés comme très encourageants et constituent, pour son secteur de recherche, un tremplin vers d'autres applications plus spectaculaires encore, visant à se rapprocher des usagers et à rendre le train et les gares plus agréables et plus « conviviaux ». Tel est le principal objectif vers lequel se dirige l'informatique.

Outre les exemples que nous avons vus, les applications de la synthèse et de la reconnaissance de la parole, envisagées par la SNCF, sont les suivantes :
— centre de renseignements téléphoniques automatiques;
— distributeurs parlants de titres de transport, avec aide vocale;
— tableaux de bord parlants pour la conduite des trains;
— commande vocale de machines-outils;
— saisie d'informations vocales pour des applications bureautiques et informatiques (ex. : liste des numéros de wagons composant un train).

Page manquante
(publicité)

Page manquante
(publicité)